CONTROL ADAPTATIVO BIOINSPIRADO UTILIZANDO CIRCUITOS DE SEÑAL MIXTA PSOC

VICTOR ANDRES ARGOTE FUERTES

UNIVERSIDAD AUTONOMA DE OCCIDENTE
FACULTAD DE INGENIERIA
DEPARTAMENTO DE AUTOMATICA Y ELECTRÓNICA
PROGRAMA DE INGENIERIA ELECTRÓNICA
SANTIAGO DE CALI
2011

CONTROL ADAPTATIVO BIOINSPIRADO UTILIZANDO CIRCUITOS DE SEÑAL MIXTA PSOC

VICTOR ANDRES ARGOTE FUERTES

Proyecto de grado para optar al título de

Ingeniero Electrónico

Director

JESÚS ALFONSO LÓPEZ

Ingeniero Electricista

UNIVERSIDAD AUTONOMA DE OCCIDENTE
FACULTAD DE INGENIERIA
DEPARTAMENTO DE AUTOMATICA Y ELECTRÓNICA
PROGRAMA DE INGENIERIA ELECTRÓNICA
SANTIAGO DE CALI
2011

Nota de aceptación:

Aprobado por el Comité de Grado en cumplimiento de los requisitos exigidos por la Universidad Autónoma de Occidente para optar al título de Ingeniero Electrónico.

JUAN CARLOS MENA

Jurado

JHON JAIRO CABRERA

Jurado

Santiago de Cali, 05 de Diciembre del 2011

CONTENIDO

LISTA DE FIGURAS

RESUMEN

En el siguiente trabajo se describe el estudio e implementación de un dispositivo electrónico que funciona como controlador adaptativo en una planta de la Universidad Autónoma de Occidente, dicho controlador se caracteriza por incorporar técnicas de computación evolutiva para el modelado y control del sistema, se utilizan algoritmos de optimización por enjambre de bacterias para tal fin.

Palabras Claves: PSoC, Control Adaptativo, Algoritmos de Optimización, Computación Evolutiva

INTRODUCCIÓN

En la industria y de forma específica en procesos llevados a cabo de manera automática, es necesario diseñar estrategias para garantizar que todos los elementos o sistemas involucrados funcionen correctamente y puedan cumplir un objetivo especifico, dichas estrategias reciben el nombre de esquemas de control, y los dispositivos encargados de garantizar el correcto funcionamiento de un sistema son llamados controladores.

Dentro de este marco contextual se suelen utilizar en su mayoría esquemas de control no siempre óptimos, que se comportan de manera uniforme durante todo su tiempo de funcionamiento. Sin embargo, sabemos que la mayoría de los sistemas no siempre se comportan igual en todos sus puntos de operación debido a variables físicas o perturbaciones externas y dicha condición no es habitualmente considerada.

Ingenieros y científicos han identificado este problema hace mucho, y desde entonces se han planteado diferentes mecanismos para controlar sistemas no lineales, es decir sistemas que no funcional igual en todos sus puntos de operación y son muy susceptibles a cambios o perturbaciones.

Las técnicas de control adaptativo son algunos de estos mecanismos de control y como su nombre lo indica se adaptan a fenómenos variables que afectan un sistema para lograr que funcione de manera adecuada.

Por otro lado, en la última década han surgido una serie de algoritmos computacionales inspirados en el comportamiento de colectividades de seres vivos como hormigas, abejas y bacterias, a los cuales se les ha denominado inteligencia de enjambres. La computación de algoritmos basados en inteligencia de enjambres ha tenido gran acogida en muchas aplicaciones donde se requiere una solución óptima a un problema, una de esas aplicaciones es el diseño de controladores inteligentes utilizando esquemas de control adaptativo.

Un inconveniente que plantea la implementación de estos algoritmos inteligentes de control, es la plataforma sobre las cual van a ejecutarse, y la elección de dicha plataforma depende en gran medida de cómo se integrara el controlador con el sistema que se desea controlar, por ejemplo, en la industria es común ver el uso

de equipos sofisticados para la ejecución de algoritmos para el control de ciertos procesos, y generalmente esto resulta siendo ineficiente debido al espacio que ocupan algunos equipos o el consumo energético de estos, es aquí donde se hace necesario pensar en un sistema integrado que permita solamente la ejecución de algoritmos para realizar una tarea específica y donde probablemente algunos componentes necesarios para su funcionamiento sean integración de algunos otros, los sistemas con estas características reciben el nombre de sistemas embebidos, y últimamente el desarrollo de dichos sistemas se ha visto impulsado por la necesidad de miniaturización de los elementos de uso cotidiano o de uso especifico, y aunque suene paradójico con el nacimiento de nuevos sistemas embebidos nacen también nuevas familias de plataformas que también son en su interior plataformas embebidas.

Durante los últimos años ha surgido una plataforma embebida muy promisoria y sobre la cual se están realizando excelentes desarrollos, esta plataforma recibe el nombre de PSoC o Sistema Programable en Chip (en inglés Programable System on Chip). Un PSoC se caracteriza por ser un circuito integrado de señal mixta, es decir un circuito integrado donde señales Analógicas y Digitales conviven conjuntamente en un mismo chip, y la ventaja que esto conlleva es la miniaturización de sistemas donde sea necesario el uso de ambas señales, un ejemplo inmediato es un controlador ya que es un dispositivo que necesita medir variables físicas de manera analógica y procesarlas de forma digital.

Teniendo en cuenta lo anterior, en este trabajo propone la implementación de algoritmos de control adaptativo en una plataforma hardware embebida y de bajo costo utilizando solamente un microchip PSoC sin ningún otro tipo de interfaz análoga-digital o digital-análoga para el desarrollo de un controlador inteligente que funcione de manera optima ante la presencia de fenómenos o perturbaciones en un sistema y donde los algoritmos de control sean adaptativos e inspirados en la inteligencia de enjambres de Bacterias.

1. MARCO TEÓRICO

Antes de iniciar con la implementación y desarrollo del sistema de control propuesto es necesario que el lector conozca ciertos términos y conceptos que serán introducidos en este capítulo, – y citados en capítulos posteriores – donde se prepara un marco teórico que comprenderá la definición de temas en las ramas del Control, la Computación Evolutiva y los Sistemas Embebidos, todas las definiciones aquí presentadas son importantes debido al papel que juegan en el diseño del controlador inteligente. Al terminar este capítulo también se habrán presentado algunos antecedentes y trabajos previos a este.

1.1. ESQUEMAS DE CONTROL

Un esquema de control es una representación de un conjunto de operaciones especiales que se realizan sobre las observaciones de las entradas y salidas de un sistema para lograr que siga características deseadas.

Generalmente cada una de las ramas de la teoría de control definen un esquema de control, así por ejemplo podemos encontrar esquemas de control clásico, moderno, adaptativo, inteligente, etc.

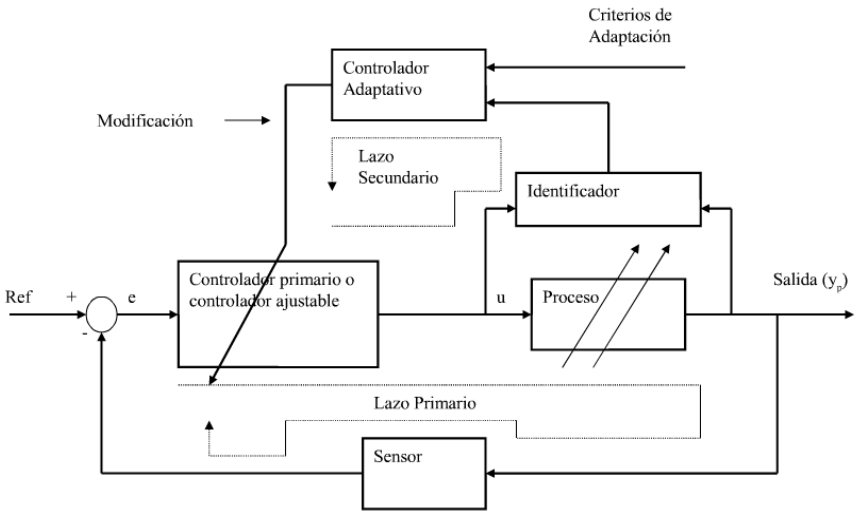
En esta sección se define el esquema de control adaptativo usado para el algoritmo que se implementara en la plataforma propuesta.

1.1.1. Control Adaptativo. El control adaptativo es un tipo especial de control en donde se adaptan los parámetros variables dentro de un sistema a fin de mantener el correcto funcionamiento del sistema [Adex, 2010].

Un sistema de control adaptativo mide las características dinámicas de una planta las compara con las características dinámicas deseadas y utiliza la diferencia para ajustar los parámetros variables del sistema de control o para ejecutar una acción que mantenga el desempeño optimo de este, independiente de las acciones

ambientales o externas. En general un esquema de control adaptativo puede ser el siguiente:

Figura 1.1 Esquema de Control Adaptativo



En este esquema de control podemos observar como el controlador adaptativo funciona como controlador secundario, de forma que modifica los parámetros del controlador primario. Este tipo de control es considerado como no lineal y es útil cuando se trabaja sobre sistemas de parámetros variables donde un controlador fijo no es el adecuado.

El mecanismo de adaptación presenta una solución en tiempo real al problema de diseño de parámetros conocidos, su forma de implementación es diversa aunque la base conceptual es la misma para todos los esquemas de control adaptativo, utilizando el libro Control Adaptativo y Robusto de Francisco Rodríguez Rubio y Manuel Jesús López Sánchez se puede profundizar aun mas sobre varios esquemas conocidos.

1.2. Computación evolutiva

En la naturaleza, los individuos compiten entre sí por recursos tales como comida, agua, refugio. Adicionalmente, los animales de la misma especie normalmente antagonizan para obtener una pareja. La computación evolutiva utiliza estos

conceptos básicos de los procesos evolutivos como una herramienta para el aprendizaje y la optimización.

Es esta sección se estudiará el funcionamiento, fundamentos teóricos, e implementación del algoritmo bioinspirado que hace parte del controlador inteligente propuesto, algunos temas como el procesamiento en paralelo y arquitectura de estos algoritmos son actualmente los paradigmas más estudiados por los investigadores que trabajan en esta disciplina.

1.2.1. Algoritmos Bioinspirados. La computación bioinspirada o algoritmos bioinspirados se basan en emplear analogías con sistemas naturales o sociales para la resolución de problemas.

Los algoritmos bioinspirados simulan el comportamiento de sistemas naturales para el diseño de métodos heurísticos no determinísticos de búsqueda, aprendizaje, comportamiento, entre otros. [Wikipedia, 2010]
En la actualidad los "Algoritmos Bioinspirados" son uno de los campos más prometedores de investigación en el diseño de algoritmos. Estos se encuentran catalogados en la rama de la computación evolutiva que a su vez se define dentro del marco de la inteligencia computacional junto con las redes neuronales y los sistemas de inferencia difusa.

De forma general los algoritmos bioinspirados se basan en sistemas de inteligencia de enjambres que están constituidos típicamente de agentes simples que interactúan entre ellos y con su ambiente. Siguen reglas simples y, aunque no existe una estructura de control que dictamine el comportamiento de cada uno de ellos, las interacciones locales entre los agentes conducen a un comportamiento global complejo. Ejemplos en la naturaleza incluyen colonias de hormigas, alineamiento de aves en vuelo, comportamiento de rebaños, crecimiento bacteriano y comportamiento de cardúmenes. [Wikipedia, 2010B]

1.3. OPTIMIZACIÓN

Es un problema que se pretende encontrar una solución o conjunto de soluciones que maximice o minimice una cierta medida de calidad, conocida como función objetivo. Dentro de los conceptos más destacados para el término optimización se encuentran:

- Proceso de buscar la mejor solución posible a un problema, bajo ciertas circunstancias [Reyes 2006].

- Método matemático para determinar los valores de las variables que hacen máximo el rendimiento de un proceso o sistema.

- Búsqueda de la mejor manera de realizar una actividad.

Los métodos de optimización son conocidos como técnicas de programación matemática, son aplicables a problemas de toma de decisiones, generalmente obtienen las mejores soluciones posibles en un tiempo considerablemente bajo. Los problemas de optimización pueden clasificarse de acuerdo a las siguientes características [Deb 1995]:

- Basándose en la existencia de restricciones
  - Con restricciones
  - Sin restricciones

- Basándose en la naturaleza de las variables de decisión
  - Problemas estáticos o paramétricos
  - Problemas dinámicos o de trayectorias (las variables son función de un parámetro determinado)

- Clasificación basada en la naturaleza de las ecuaciones involucradas
  - Problemas lineales
  - Problemas no lineales
  - Problemas de programación geométrica
  - Problemas de programación cuadrática

- Basándose en los valores permisibles de las variables de diseño
  - Problemas de programación entera
  - Problemas de programación con valores reales
  - Cuando se busca un orden optimo de elementos a optimizarse, hablamos de un problema de optimización combinatoria.

- Basándose en la naturaleza determinista de las variables
  - Problemas estocásticos
  - Problemas deterministas

- Basándose en la separabilidad de las funciones
  - Problemas separables

    - Problemas no separables

- Basándose en el número de funciones objetivo
    - Mono-objetivo
    - Multi-objetivo

El tipo de problema de optimización que se abordara en este trabajo es el problema de programación no lineal y que caracterizan a los problemas de diseño de controladores adaptativos.

1.3.1. Optimización por Colonias Hormigas (ACO). Basados en la simulación del comportamiento de las colonias de Hormigas cuando recogen comida. Se pueden mencionar algunos comportamientos asombrosos de las Hormigas como se puede ver en [Gomer, 2006] y son los siguientes:

- El 50 % de los insectos sociales son Hormigas.

- La presencia en la tierra de estos insectos es de 10^8 años con alrededor de 11880 especies distintas.

- Constituyen un 15 % de la Biomasa del Bosque tropical.

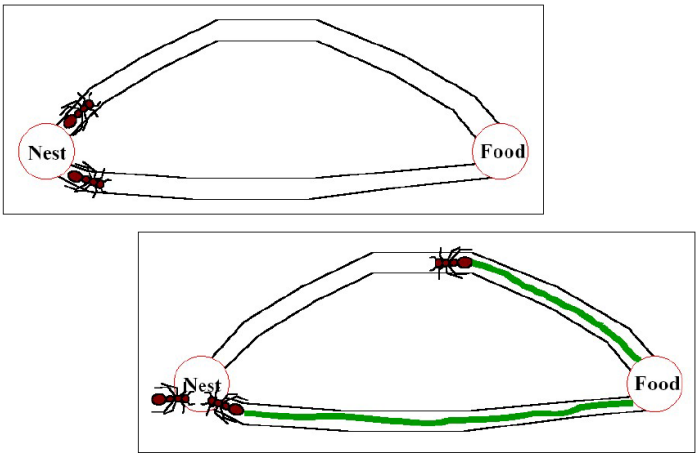
- Han Colonizado casi toda la masa terrestre.

- Una Colonia opera como si fuese una sola entidad (Superorganismo).

Estas características hacen de las hormigas un buen modelo para resolver problemas en computación, los cuales al abordarlos por los algoritmos tradicionales, no se encuentra una solución, o la encontrada no es buena. De esta forma se podría comprobar la potencia de estos algoritmos resolviendo problemas complejos, por ejemplo el problema del agente viajero.

La técnica en computación para la utilización de algoritmos inspirados en la forma de supervivencia de las hormigas fue denominada como Ant Colony Optimization (ACO). Es una forma de simular el comportamiento de las hormigas cuando salen en busca de alimento y de esta forma tratar de dar solución a algunos problemas de la vida real. La técnica consiste en resolver problemas que pueden ser transformados a encontrar buenos caminos sobre grafos.

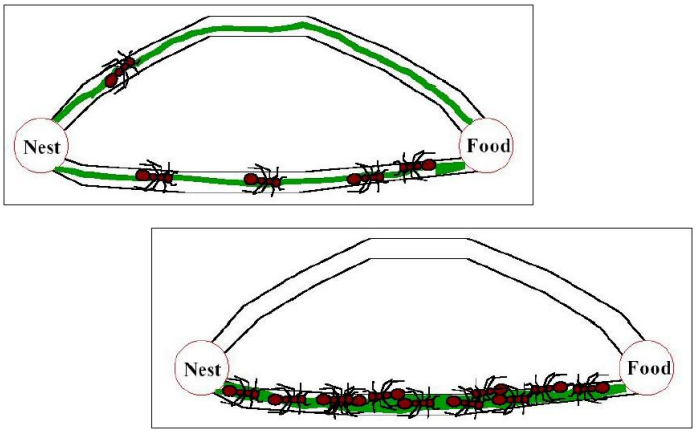
Figura 1.2 Comienzo del proceso de las hormigas en busca de Alimento



Fuente: [Gomer, 2006].

Inicialmente, las hormigas eligen al azar distintos caminos como se muestra en la Figura 1.3.1.1. y dejan rastros de feromonas por donde se mueven. Como se muestra también en la Figura 1.3.1.1., hubo una hormiga que hizo más rápido el recorrido en la búsqueda de alimento, esto hace que las otras hormigas sigan el rastro de feromona de ella y salgan por ese mismo camino. Las hormigas que van saliendo del nido, siguen el camino en el cual haya más rastro de feromona como se muestra la siguiente figura.

Figura 1.3 El camino más corto



Fuente: [Gomer, 2006]

Finalmente las hormigas encuentran el camino más corto desde su nido hasta la comida. De esta forma, todas las hormigas siguen este camino para recolectar el alimento del enjambre y suplir la necesidad de alimento.

En computación el funcionamiento es similar [Gomer, 2006]: se crean hormigas artificiales que imitan el comportamiento de las hormigas reales. Se mueven por el grafo depositando feromonas de manera que las exploradoras posteriores construirán mejores soluciones; es decir, se trata de solucionar problemas difíciles en optimización combinatoria. Este modelo es muy útil para solucionar problemas de enrutamiento en redes de Telecomunicaciones.

"*Otras características usualmente incluidas dentro de un sistema ACO son el balance entre las características aleatorias y la severidad en la toma de decisiones (exploración/ explotación del conocimiento); y estrategias de conocimiento global denominadas daemon actions. Las características de un algoritmo ACO pueden necesitar alguna clase de sincronización, usualmente obtenida por una programación secuencial, donde el conocimiento global es fácilmente accesible en cualquier instante*" [Muñoz, 2008].

1.3.2. Optimización por Colonias de Abejas (BCO). Otra de las técnicas importantes de los algoritmos Bioinspirados son las Abejas. Algunos comportamientos interesantes de las abejas se muestran en [Herrera, 2008]. Algunas de las características de las abejas que pueden ser usadas como modelos de comportamiento inteligente son: la división de labores, la comunicación sobre un individuo y grupos, el comportamiento cooperativo y su estrategia reproductiva. Estos comportamientos realizan una secuencia de acciones producto de su potencialidad genética, ambientes ecológicos y fisiológicos, las condiciones sociales de la colonia y otras prioridades.

Un comportamiento que se puede modelar de forma algorítmica es el sistema de búsqueda de néctar. Nos habla que tal vez lo más particular de este comportamiento es la forma como las abejas se comunican a través de danzas para informar posibles fuentes de alimento a toda la colonia [Muñoz, 2007].
También menciona que la colonia de abejas tiene que adaptarse continuamente a cada nueva situación, procurando tener la cantidad de néctar necesario para el mantenimiento de la colmena, por medio de la división apropiada de la fuerza de

trabajo entre la exploración del campo de nuevas fuentes de alimento y la explotación de las fuentes conocidas.

El comportamiento *foraging* de cada abeja como individuo, es regulada y la información interna y externa juegan roles importantes en el comportamiento del *foraging*. Las reglas incluyen especificaciones para el viaje de partida de la abeja del nido a la fuente, la búsqueda de la fuente para una abeja, la recolección de néctar de una fuente, el viaje de regreso al nido, el medio por el cual la información acerca de la fuente sea transmitida a otras abejas en el nido.

La información representada por las danzas indica la localización de las fuentes en la distancia (por medio de la duración), la posición (por medio de la dirección) y la productividad de la fuente (por medio de la insistencia y vigorosidad del baile). Entre más vigorosos y alargados sean los bailes, más seguidores podrían ser atraídos. Siguiendo el modelo Biológico, el artículo implementa un algoritmo de búsqueda de Néctar modificando algunas de sus características para efectuar algunas tareas de una manera más eficiente de tal manera que se pueda utilizar como algoritmo de optimización, haciendo uso de la información local, la transmisión de información por medio de danzas y un balance entre la exploración y la explotación.

[Ziad, 2009] El algoritmo inicializa la población con un número determinado de Abejas, evalúa el Fitness (la calidad de la población). Después se deben seleccionar algunos sitios vecinos para visitarlos, luego determinar el tamaño de sitios vecinos (tamaño del parche) con algún tipo de provecho. Ahora se deben enviar abejas a los sitios seleccionados (más abejas a los mejores sitios). En este punto se debe seleccionar la abeja más apta de cada sitio. Se asignan las abejas (la operación entre las abejas del enjambre menos los sitios seleccionados) para que realicen búsquedas aleatorias a otros sitios. Después de esto se crea una nueva población a partir de las mejores abejas, de esta forma se asegura que a la próxima generación llegarán las mejores. Este ciclo se repetirá cuantas veces se haya programado.

1.3.3. Optimización por Enjambre de Partículas (PSO). Es una técnica de optimización inspirada en el comportamiento social de bandadas de aves o peces. Comparte muchas similitudes con las técnicas de computación evolutiva, tales como Algoritmos Genéticos (AG). El sistema se inicia con una población al azar y la búsqueda de buenas soluciones mediante la actualización de las generaciones. Sin embargo, a diferencia de AG, PSO no tiene evolución, tales como los operadores de cruce y mutación. En PSO, las soluciones potenciales, llamadas

partículas, vuelan por el espacio del problema siguiendo el curso de partículas óptimo.

La PSO cuenta con una población de entidades llamadas “partículas”, la cual es inicializada y luego, con base en una función de evaluación (función objetivo), se determina cuán bueno es cada individuo. En un proceso iterativo, la heurística afina gradualmente la dirección que las partículas deben seguir de forma de encontrar la zona del espacio de búsqueda donde se encuentran buenas soluciones. Para este ajuste de direcciones se tiene en cuenta la propia experiencia de cada partícula y la experiencia de sus vecinos. Dependiendo del tamaño de vecindario que se considere existen dos modelos básicos de PSO: El algoritmo PSO global (*gbest*) que considera como vecindario de cada partícula a la totalidad del swarm y el algoritmo PSO local (*lbest*) que permite definir distintas topologías de vecindarios de menor tamaño para cada partícula del swarm [Aragón, 2006].

1.3.4. Optimización por Enjambre de Bacterias (BFO). Las bacterias son minúsculas nanomáquinas poderosamente adaptadas para formar redes de proceso masivamente paralelo (realmente masivo) y descentralizado. Entidades de red móviles, de código genético reducido, especializadas pero al mismo tiempo adaptables a cambios imprevistos, autoreproducibles a una velocidad frenética y tolerantes a fallos [Blissett, 2002].

Lynn Margulis y otros hablan de las comunidades bacterianas como una red global de intercambio genético a escala planetaria que ha persistido durante miles de millones de años y que si se trata de traducir eso a términos informáticos, se pueden caracterizar las comunidades bacterianas como redes de código abierto en clave genética, más allá de la pura analogía.

Este tipo de optimización representa una aproximación diferente a la búsqueda de valores óptimos en funciones no lineales, basado en el comportamiento quimiotáctico de la E. Coli. Si bien utilizar la quimiotaxis como modelo para optimización hace unos años, el trabajo de Passino incluyó algunas modificaciones como la reproducción y la dispersión de los agentes [Muñoz, 2008]. La *E. Coli* es tal vez el microorganismo más comprendido, ya que su comportamiento y estructura genética están bien estudiados. Esta consta de una cápsula que incluye sus órganos, y flagelos que utiliza para su locomoción; posee capacidad de reproducirse por división y también es capaz de intercambiar información genética con sus congéneres. Además, puede detectar alimento y evitar sustancias nocivas, efectuando un tipo de búsqueda aleatoria, basado en dos estados de locomoción, el desplazamiento y el giro. La decisión de

permanecer en uno de estos dos estados se debe a la concentración de nutrientes o sustancias nocivas en el medio. Este comportamiento se denomina quimiotaxis [Muñoz, 2008].

BFO incluye cuatro mecanismos fundamentales: quimiotaxis (*nado y giro*), agrupamiento (*Swarming*), reproducción, y eliminación-dispersión. Este modelo se ha utilizado con éxito para la sintonización de controladores adaptativos, la sincronización de controladores PID, estudio de estabilidad y la regulación de temperatura en superficies. También se ha utilizado como hibridación con algoritmos genéticos para mejorar su desempeño [Muñoz, 2008].

Finalmente se puede decir que es una buena técnica y que puede llegar a solucionar problemas complejos de optimización. Algunas aplicaciones de esta técnica se mencionan en [Muñoz, 2008], por ejemplo, la sintonización de controladores adaptativos, la sintonización de controladores PID y la regulación de temperatura en superficies.

1.4. CIRCUITOS DE SEÑAL MIXTA

Los circuitos de señal mixta son circuitos integrados que contienen circuitos analógicos y digitales combinados en un solo semiconductor.

Hasta mediados de los años 90, se trataban generalmente de circuitos integrados para conversión analógica-digital, conversión digital-analógica, módems, fuentes de alimentación electrónica o circuitos integrados de búfer digital. Hoy en día los podemos encontrar casi en cualquier dispositivo electrónico incluidos los circuitos de sonido digital controlados por circuitos de señal mixta y con el nacimiento de las tecnologías celular y de redes en los teléfonos celulares o móviles, emisiones de radio por software y routers WAN y Red de área local (LAN).

Unos de los desafíos particulares de los circuitos de señal mixta es probar el funcionamiento correcto de los circuitos ya que es complejo, costoso y a menudo debe realizarse de uno en uno.

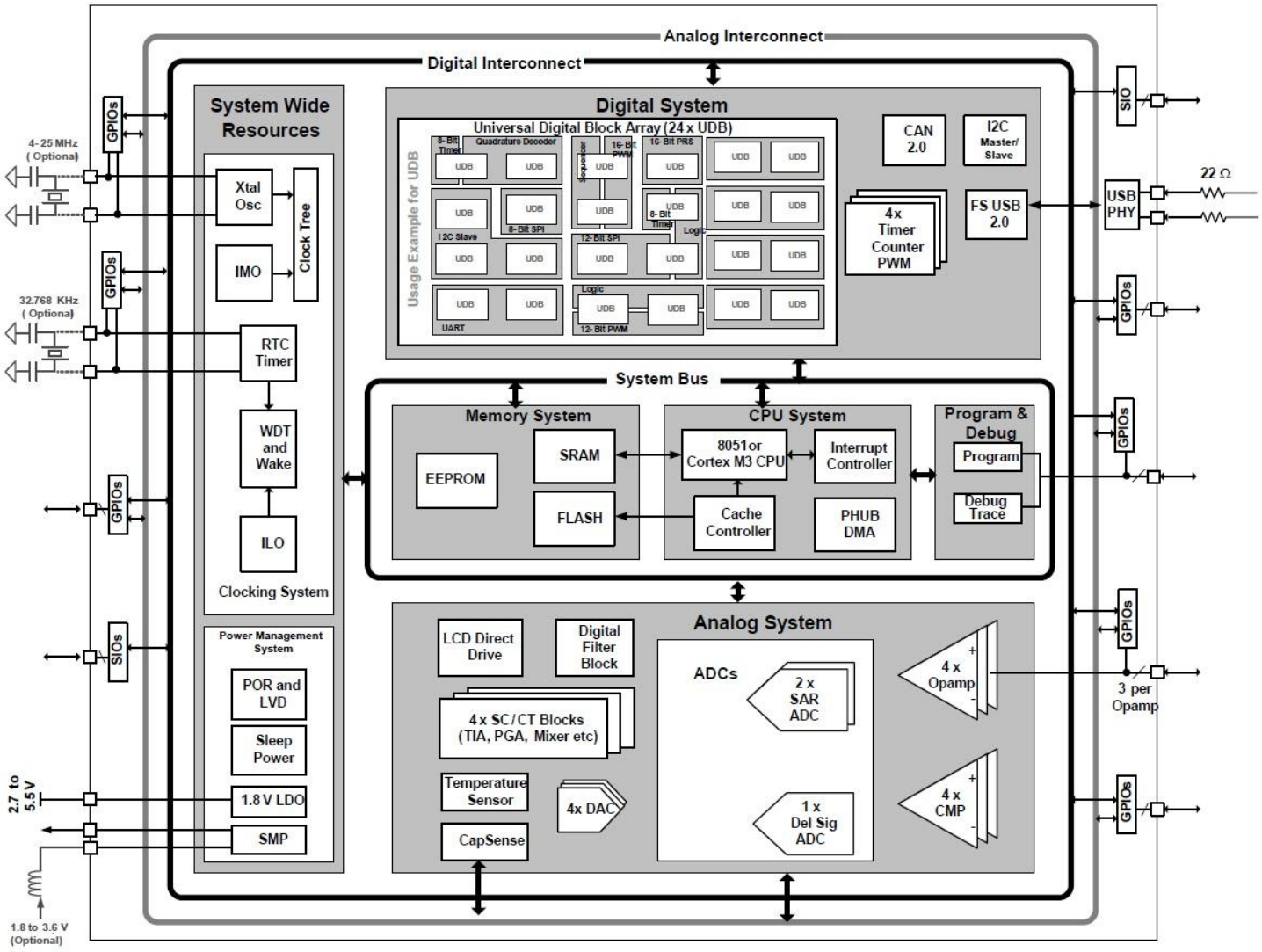
Generalmente, el diseño analógico de circuitos no puede ser automatizado al nivel que se consigue en los circuitos digitales, combinar las dos tecnologías multiplica esta complicación y los costos de estos chips es un tanto más alto que el de chips netamente digitales. A continuación veremos un tipo de arquitectura de señal mixta llamada PSoC la cual utilizaremos para gestionar toda la parte electrónica del controlador inteligente.

1.4.1. Circuito de Señal Mixta PSoC. PSoC son las siglas de Programable System On Chip, fue desarrollado por la empresa Cypress Semiconductor en 2002.

Se caracteriza porque incorpora una CPU, y arreglos de señal mixta configurables a través de registros en forma de bloques análogos y digitales lo que posibilita la reducción de componentes externos y además permite la reconfiguración en línea de todos sus bloques para utilizar por ejemplo en procesos adaptativos.
El circuito integrado de señal mixta PSoC comprado con otras familias de microcontroladores, es realmente un dispositivo muy versátil, a la fecha (Abril 2010) existen tres plataformas PSoC delimitadas por el tipo de procesador embebido. [Wikipedia, 2010A]

- PSoC1 (CY8C2xxxx): CPU M8C
- PSoC3 (CY8C3xxxx): CPU 8051
- PSoC5 (CY8C5xxxx): CPU ARM Cortex M3

Figura 1.4 Diagrama simplificado de la arquitectura PSoC 5



Fuente: Cypress PSoC 3/5 TRM

Cada una tiene ventajas y desventajas, para nuestro caso se hace necesario el uso del PSoC 5 ya que es la arquitectura con mayor capacidad de procesamiento y sobre la cual ya se encuentra implementado el sistema operativo de tiempo real FreeRTOS que se describirá más adelante.

Como podemos observar de la figura anterior, el PSoC contiene bloques análogos y digitales independientes del sistema de procesamiento, lo que nos brinda un grado más de paralelismo y grandes ventajas a la hora de implementar un dispositivo con muchas tareas ejecutándose al mismo tiempo.

1.5. SISTEMAS OPERATIVOS EN TIEMPO REAL

SOTR o RTOS -"Real Time Operating System" en inglés, es un sistema operativo que ha sido desarrollado para aplicaciones de tiempo real. Como tal, se le exige corrección en sus respuestas bajo ciertas restricciones de tiempo. Si no las respeta, se dirá que el sistema ha fallado. Para garantizar el comportamiento correcto en el tiempo requerido se necesita que el sistema sea predecible (determinista).

Es usado típicamente para aplicaciones integradas, normalmente tiene las siguientes características:

- No utiliza mucha memoria
- Cualquier evento en el soporte físico puede hacer que se ejecute una tarea
- Multi-arquitectura (el código puede ser portado a cualquier tipo de CPU)

Se caracterizan por presentar requisitos especiales en cinco áreas generales:

- Determinismo
- Sensibilidad
- Control del usuario

- Fiabilidad
- Tolerancia a los fallos

Este tipo de sistemas operativos no es necesariamente eficiente en el sentido de tener una capacidad de procesamiento alta. El algoritmo de programación especializado, y a veces una tasa de interrupción del reloj alta pueden interferir en la capacidad de procesamiento.

Aunque para propósito general un procesador moderno suele ser más rápido, para programación en tiempo real deben utilizarse procesadores lo más predecibles posible, sin paginación. Todos estos factores añaden una aleatoriedad que hace que sea difícil demostrar que el sistema es viable, es decir, que cumple con los plazos.

Hay dos diseños básicos para sistemas operativos en tiempo real:

- Un sistema operativo guiado por eventos sólo cambia de tarea cuando un evento necesita el servicio.
- Un diseño de compartición de tiempo cambia de tareas por interrupciones del reloj y por eventos.

El diseño de compartición de tiempo gasta más tiempo de la CPU en cambios de tarea innecesarios. Sin embargo, da una mejor ilusión de multitarea. Normalmente se utiliza un sistema de prioridades fijas.

En los diseños típicos, una tarea tiene tres estados: ejecución, preparada y bloqueada. La mayoría de las tareas están bloqueadas casi todo el tiempo. Solamente se ejecuta una tarea por CPU. La lista de tareas preparadas suele ser corta, de dos o tres tareas como mucho.

En un sistema operativo en tiempo real bien diseñado, preparar una nueva tarea necesita de 3 a 20 instrucciones por cada entrada en la cola y la restauración de la tarea preparada de máxima prioridad de 5 a 30 instrucciones.

En un procesador 68000 20MHz, los tiempos de cambio de tarea son de 20 microsegundos con dos tareas preparadas.

Cientos de CPUs MIP ARM pueden cambiar en unos pocos microsegundos.
Las diferentes tareas de un sistema no pueden utilizar los mismos datos o componentes físicos al mismo tiempo. Hay dos métodos para tratar este problema. Uno de los métodos utiliza semáforos. En general, el Semáforo binario puede estar cerrado o abierto. Cuando está cerrado hay una cola de tareas esperando la apertura del semáforo, cuando el semáforo se abre pueden utilizar un dato y se cierra el semáforo para que otras tareas no puedan acceder a él.

La otra solución es que las tareas se manden mensajes entre ellas y es la forma como en el controlador inteligente se transmite la información dentro de cada uno de los bloques que forman el esquema de control bioinspirado.

Las interrupciones son la forma más común de pasar información desde el mundo exterior al programa y son, por naturaleza, impredecibles. En un sistema de tiempo real estas interrupciones pueden informar diferentes eventos como la presencia de nueva información en un puerto de comunicaciones, de una nueva muestra de audio en un equipo de sonido o de un nuevo cuadro de imagen en una videograbadora digital.

Para que el programa cumpla con su cometido de ser tiempo real es necesario que el sistema atienda la interrupción y procese la información obtenida antes de que se presente la siguiente interrupción. Como el microprocesador normalmente solo puede atender una interrupción a la vez, es necesario que los controladores de tiempo real se ejecuten en el menor tiempo posible. Esto se logra no procesando la señal dentro de la interrupción, sino enviando un mensaje a una tarea o solucionando un semáforo que está siendo esperado por una tarea. El programador de tareas dentro del sistema operativo se encarga de activar la tarea y esta se encarga de adquirir la información y completar el procesamiento de la misma.

Las interrupciones serán utilizadas en el controlador inteligente para atender la interfaz de acceso a él.

1.5.1. FreeRTOS. FreeRTOS es un sistema operativo de tiempo real para dispositivos embebidos, ha sido portado a muchas arquitecturas incluyendo PSoC 5. Es distribuido bajo la licencia GPL con una excepción y es que se le permite al

usuario mantener el código fuente de la aplicación cerrada, facilitando su uso para aplicaciones comerciales.

FreeRTOS ha sido diseñado para ser pequeño y simple. El kernel o núcleo por si mismo consiste solamente en tres o cuatro archivos en C. Para hacer el código legible, fácil de portear, y mantenible ha sido escrito en C, pero hay algunas instrucciones en Assembler específicas para cada procesador.
Las características principales son:

- Pequeño y simple. Muy bueno para quienes son nuevos en el uso de sistemas operativos.
- El programador de tareas puede ser preemptivo o cooperativo.
- Soporte para coroutine en FreeRTOS es simple y liviano, cada tarea tiene un uso limitado del stack.

Las siguientes son algunas de las arquitecturas soportadas por FreeRTOS:

- Altera Nios II
- ARM7, ARM9, ARM Cortex M3
- AVR, AVR32
- Cortus - APS3
- Fujitsu
- Freescale
- Coldfire V1
- Coldfire V2
- Intel x86, Intel 8051
- PIC18, PIC24, dsPIC, PIC32

- SuperH
- V850
- MSP430
- Xilinx
- MicroBlaze

1.6. ESTRUCTURAS DE DATOS

En programación, una estructura de datos es una forma de organizar un conjunto de datos elementales con el objetivo de facilitar su manipulación. Un dato elemental es la mínima información que se tiene en un sistema.
Una estructura de datos define la organización e interrelación de éstos y un conjunto de operaciones que se pueden realizar sobre ellos. Las operaciones básicas son:

- Alta, adicionar un nuevo valor a la estructura.
- Baja, borrar un valor de la estructura.
- Búsqueda, encontrar un determinado valor en la estructura para realizar una operación con este valor, en forma secuencial o binario (siempre y cuando los datos estén ordenados).
- Otras operaciones que se pueden realizar son:
- Ordenamiento, de los elementos pertenecientes a la estructura.
- Apareo, dadas dos estructuras originar una nueva ordenada y que contenga a las apareadas.

Cada estructura ofrece ventajas y desventajas en relación a la simplicidad y eficiencia para la realización de cada operación. De esta forma, la elección de la estructura de datos apropiada para cada problema depende de factores como la frecuencia y el orden en que se realiza cada operación sobre los datos.
Estos son algunos ejemplos de Estructuras de datos:

- Vectores (matriz o arreglo)
- Registro
- Tipo de datos algebraico
- Listas Enlazadas
- Pila (stack)
- Colas (queue)
- Árboles
- Conjuntos (set)
- Grafos
- Tablas "Hash"
- *Montículos (o heaps)

En este trabajo nos centraremos en la descripción le las colas o Queues como estructuras de datos.

1.6.1. Queues. Una cola en inglés Queue es una estructura de datos, caracterizada por ser una secuencia de elementos en la que la operación de inserción "push" se realiza por un extremo y la operación de extracción "pop" por el otro. También se le llama estructura FIFO (del inglés "First In First Out"), debido a que el primer elemento en entrar será también el primero en salir.

Las colas se utilizan en sistemas informáticos, transportes y operaciones de investigación (entre otros), dónde los objetos, personas o eventos son tomados como datos que se almacenan y se guardan mediante colas para su posterior procesamiento.

Este tipo de estructura de datos abstracta se implementa en Programación orientada a objetos mediante clases, en forma de listas enlazadas.

La particularidad de una estructura de datos de cola es el hecho de que sólo podemos acceder al primer y al último elemento de la estructura. Así mismo, los elementos sólo se pueden eliminar por el principio y sólo se pueden añadir por el final de la cola.

Ejemplos de colas en la vida real serían: personas comprando en un supermercado, esperando para entrar a ver un partido de béisbol, esperando en el cine para ver una película, una pequeña peluquería, etc. La idea esencial es que son todas líneas de espera.

En caso de estar vacía la cola, borrar un elemento sería imposible hasta que no se añade un nuevo elemento.

Las operaciones básicas para una Cola son:

- Crear: se crea la cola vacía.
- Encolar (añadir, entrar, push): se añade un elemento a la cola. Se añade al final de esta.
- Desencolar (sacar, salir, pop): se elimina el elemento frontal de la cola, es decir, el primer elemento que entró.
- Frente (consultar, front): se devuelve el elemento frontal de la cola, es decir, el primero elemento que entró.

Como se ha comentado anteriormente cada bloque del controlador inteligente es una tarea independiente en el sistema operativo y las colas son las estructuras de datos que permiten a cada bloque emitir y recibir señales de los otros bloques.

1.7. IDENTIFICACIÓN

La identificación de sistemas puede definirse como el área de la teoría de sistemas que estudia metodologías para la obtención de modelos matemáticos de sistemas dinámicos a partir de datos de medición de las entradas y salidas del sistema. La identificación de sistemas se ha convertido en una herramienta fundamental en muchas ramas de la ingeniería y otras áreas, que requieren de la

existencia de modelos precisos del sistema que posibiliten el análisis, simulación, el diseño e implementación de estrategias de control [Gomez, 2011].

En aplicaciones de Control, la obtención de un modelo matemático más o menos preciso del sistema es fundamental ya que la mayoría de los métodos de diseño de controladores parten de la hipótesis de que un modelo parametrizado del proceso está disponible. Estos modelos necesitan simular el comportamiento real en los casos en que existe un conocimiento previo limitado de la estructura del sistema.

Consideremos el motor de un coche por ejemplo. Es importante simular el comportamiento de un motor para la detección de fallos y para propósitos de diagnosis. Algunos factores importantes que afectan este proceso son las dificultades asociadas con el desarrollo de un modelo aceptable con un orden de complejidad mínimo y con un número de medidas mínimo. Estos factores hacen que el proceso de modelado de un motor de coche bastante difícil. Esto puede ser generalizado a una amplia clase de sistemas en la industria.

Dentro de los modelos utilizados para la identificación de sistemas podemos encontrar dos categorías muy importantes, los modelos paramétricos y los no-parametricos.

Dentro de los modelos no-paramétricos podemos encontrar los siguientes:

- **Análisis de Transitorios:** Este modelo usa la respuesta del sistema(overshoot, settling time, rise time) a una función impulso o paso. Util para sistemas de 1 y 2 orden.

- **Análisis de Frecuencia:** Se aplican señales senoidales de diferente frecuencia a la entrada y se hace una representación gráfica del sistema contra la fase y amplitud de salida(e.g. diagrama de Bode).

- **Análisis de Correlación:** Basado generalmente en entradas de ruido blanco. La relación salida-entrada del sistema se basa en el análisis de autocovarianza y covarianza cruzada.

- **Análisis Espectral:** Este método puede ser usado con cualquier entrada arbitraria, la relación entrada-salida se obtiene mediante un diagrama de bode

o similar. También utiliza el análisis de correlación. Desarrollado a partir de métodos estadísticos.

Y dentro de los modelos paramétricos los siguientes:

- **Modelo ARX:** Son la primera elección en un procedimiento de identificación de sistemas lineales.
- **Modelo ARMAX:** Describe el error en la ecuación como un promedio móvil.
- **Modelo Output Error(OE) :** Es un modelo ARMAX con relación in/out sin perturbación más ruido blanco aditivo en la salida.
- **Modelo Box-Jenkins(BJ) :** Es una generalización del modelo output error.

1.7.1. Modelo ARX. Un modelo auto-regresivo tipo ARX es un modelo discreto lineal en el que la salida en un instante de muestreo n se obtiene a partir de valores pasados de la salida y de la entrada. Como ejemplo de modelo autorregresivo se puede proponer el siguiente,

Ecuacion 1.7.1.1. [Gomez, 2011]

$$y_n = a_1 y_{n-1} + a_2 y_{n-2} + b_1 u_{n-1}$$

Los parámetros de un modelo lineal ARX como el anterior se determinan a partir de valores muestreados de la entrada(u) y de la salida(y) en un intervalo de tiempo T. Esta determinación se realiza mediante regresión lineal multivariable, esto es, se determinan los parámetros a_1, a_2 y $b_1$que minimizan la diferencia cuadrática entre los valores reales y los calculados por el modelo. Pueden definirse también mas parámetros hasta a_n y b_n pero esto hace más complicado estimarlos.

1.8. ANTECEDENTES

Los algoritmos bioinspirados son un marco de estudio relativamente reciente, han sido desarrollados en base a estudios de los comportamientos de organismos vivos frente a situaciones búsqueda de alimentos o recursos para su supervivencia. El trabajo que se pretende hacer está basado principalmente en los estudios de Kevin Passino quien propuso la teoría de la optimización por

enjambres de bacterias en el año 2002, y en estudios sobre la teoría de optimización por enjambre de partículas.

Las aplicaciones de estas técnicas de control han sido probadas y aplicadas a nivel software, pero el campo de implementación hardware ha sido poco estudiado.

La optimización por enjambres de bacterias es un caso ya implementado a nivel software, el primer algoritmo conocido fue desarrollado por Kevin Passino [Passino, 2002], pero presentaba inconvenientes computacionales, debido a la carga que estos algoritmos representan para una CPU, un sistema de naturaleza secuencial. Se han presentado hasta el momento varias optimizaciones del código haciéndolo más eficiente, pero aun el algoritmo es lento comparado con otros métodos de optimización bioinspirados, aunque esto no deteriora la convergencia.

Sobre implementaciones a nivel de hardware del algoritmo de optimización por enjambre de bacterias no se tiene información, pero se deduce que implementación a este nivel reduce la carga computacional y por tanto los costos que estos conllevan, haciendo posible la creación de sistemas embebidos que puedan utilizar el algoritmo para llevar a cabo algún tipo de optimización avanzada.

Se conoce también que el algoritmo de optimización ha sido aplicado con éxito para realizar controladores adaptativos, se tienen algunos esquemas de control posibles donde el algoritmo tiene funcionalidades específicas que sirven de base para la toma de decisiones del controlador.

El nacimiento de los circuitos de señal mixta PSoC también es relativamente nuevo, surgieron en el 2002, el mismo año en que Kevin Passino publico su trabajo sobre optimización inspirado en bacterias. Se conocen algunas pocas publicaciones en las que el PSoC hace operaciones de control clásico muy sencillas, hasta el momento no se tiene información sobre ningún tipo de control adaptativo o algoritmo bioinspirado embebido dentro de un PSoC, de todas formas tales tareas son difíciles de realizar en los circuitos integrados PSoC1 que eran los únicos existentes hasta mediados de 2009. Luego de la salida al mercado de las arquitecturas PSoC3 y PSoC5, se vislumbraba que tales tareas eran ahora alcanzables ya que las velocidades de los núcleos de estas arquitecturas superaban en más de cuatro veces el PSoC1. A la fecha Septiembre 2011 solo el PSoC3 es comercializado, el desarrollo de este trabajo se hace sobre la arquitectura PSoC5 ya que es la única para la que existen ports del núcleo del

sistema operativo FreeRTOS hasta el momento, de haber disponible un port de FreeRTOS para la arquitectura PSoC3 sería posible ejecutar los algoritmos sin necesidad de modificar el código ya que todas las funciones en el código son llamadas al núcleo de FreeRTOS.

2. PROPUESTA DE SOLUCIÓN

Ahora que ya hemos delimitado nuestra área de trabajo para la realización del controlador inteligente veremos cómo utilizando todas las herramientas teóricas vistas en el capitulo anterior se procede con la implementación de todo el sistema en una plataforma hardware. En este capítulo definiremos los objetivos que se desean alcanzar para el desarrollo de este trabajo, plantearemos el problema que se quiere resolver y luego se genera la propuesta completa para resolver el problema planteado y obtener así un controlador inteligente basado en algoritmos de optimización por enjambre de bacterias utilizando circuitos de señal mixta PSoC 5.

2.1. OBJETIVO GENERAL

- Realizar una implementación de un esquema de control adaptativo con algoritmos bioinspirados sobre un circuito de señal mixta PSoC la cual será validada sobre uno de los procesos de laboratorio de automática.

2.2. OBJETIVOS ESPECÍFICOS

- Estudiar los algoritmos de inteligencia de enjambres.
- Comprender los diferentes esquemas de control adaptativo.
- Seleccionar el algoritmo de inteligencia de enjambres y esquema de control adaptativo a utilizar**.**
- Evaluar y seleccionar la arquitectura PSoC que se ajuste mejor para la implementación del algoritmo de control bioinspirado.
- Diseñar e implementar el esquema de control adaptativo en la plataforma PSoC.
- Validar la implementación aplicando el controlador implementado, sobre uno de los procesos del laboratorio de Automática.

2.3. ALGORITMO DE OPTIMIZACIÓN BASADO EN BACTERIAS

En esta sección se describen los elementos que intervienen en el algoritmo de optimización basado en bacterias. Se describirá su funcionamiento y el pseudocódigo para su implementación.

2.3.1. Bacteria E.Coli. E. Coli, es un tipo común de bacteria con un grosor de 1um y un largo de aproximadamente 2um, y bajo condiciones apropiadas puede reproducirse en 20 minutos. Una E. Coli va alternando sus movimientos entre nadar o girar.

Estas bacterias tienen flagelos que funcionan como hélices que permiten a la bacteria moverse, si nadan, las hélices van en sentido contrario de las manecillas del reloj, si giran, los flagelos van en sentido de las manecillas del reloj. Acciones quimio tácticas:

- Si esta dentro de un medio neutro de nutrientes (alterna nados y giros), hace la búsqueda de nutrientes.
- Si esta nadando en un espacio de nutrientes (o fuera de sustancias nocivas) y nada en ese espacio (la concentración de nutrientes sube y los sustancias nocivos disminuyen), busca ambientes cada vez más favorables.
- Si esta nadando en un espacio de nutrientes (o arriba de sustancias nocivas), entonces busca y evita ambientes desfavorables.

De esta manera, las bacterias pueden escalar colinas de nutrientes y evitar sustancias nocivas al mismo tiempo.

Las E. Coli segregan quimioatrayentes que perciben a través de sus sensores para comunicarse entre la población de bacterias y como receptores de proteínas (un pequeño cambio en la concentración de nutrientes puede causar un significativo cambio en el comportamiento).

Los sensores hacen un promedio de las concentraciones detectadas en un lapso de tiempo.

2.3.2. Descripción del Algoritmo. El algoritmo propuesto por K.M. Passino en 2002 recibe el nombre en inglés de Bacterial Foraging Optimization Algorithm o BFOA, simula el proceso completo de forrajeo o búsqueda de nutrientes de las bacterias E. Coli como movimientos de giro y nado, reproducción y eliminación-dispersión.

El modelado de la búsqueda de alimento de las bacterias mediante el BFOA, se lleva a cabo de la siguiente manera:

- Inicialmente, las bacterias son distribuidas al azar sobre un espacio de nutrientes, que en la implementación se refiere a la llamada función de costo, que es una medida del error que se desea optimizar.

- En la 1ra. generación, las bacterias (E. Coli) se mueven hacia los puntos de concentración de nutrientes (dentro de esta generación ocurre una eliminación y dispersión), algunas bacterias se acercan a puntos de sustancias nocivas, después ocurre la reproducción y posteriormente casi todas las bacterias se colocan en puntos de concentración de nutrientes.

- En la 2da. generación, las bacterias han encontrado los puntos de concentración de nutrientes, aunque no todos estos puntos son óptimos, es decir, los más altos de concentración de nutrientes (las bacterias se enfrentan a los ruidos de los puntos los cuales no permiten que las bacterias encuentren el mejor punto, pero estas se agrupan, utilizando la comunicación mediante segregación de atrayentes y realizan la búsqueda juntas logrando casi el optimo).

- En la 3ra. generación, las bacterias encuentran el punto más alto de concentración de nutrientes.

- En la 4ta. generación, las bacterias permanecen en ese punto y la búsqueda de alimento ha terminado; algunas se dispersan o eliminan y buscan una nueva fuente de nutrientes, en algoritmo esta etapa es la re-ejecución de todos los pasos anteriores pero con la última posición guardada de cada bacteria.

Durante estas generaciones las bacterias se enfrentan a varias problemáticas en la búsqueda de sus alimentos, por ejemplo: ambientes nocivos que deben detectar y retroceder en su viaje, que causan la eliminación y dispersión (para ello estas bacterias se agrupan y son capaces de producir y secretar sustancias químicas que son usadas como mecanismo de señalización y comunicación entre ellas), el tiempo de vida de las bacterias, el recorrido que tienen que hacer para encontrar el punto optimo de concentración de nutrientes. Usando el modelo biológico que permite encontrar este punto optimo, tomamos en cuenta estas posibles restricciones y se calcula la suma de las señales enviadas por cada una de las otras bacterias de la colonia, obteniendo así el punto deseado.

2.3.3. Pseudocódigo. Basados en estos pasos, Passino [Passino, 2010] propuso el algoritmo de optimización de bacterias en búsqueda de alimento modelando el paso quimiotactico con la generación de direcciones aleatorias de la siguiente forma (algunas variables se definen en la tabla 2.3.3.1):

Ecuación 2.3.3.1. [Passino, 2010]

$$\emptyset(i) = \frac{\Delta(i)}{\sqrt{\Delta(i)^T \Delta(i)}}$$

Donde $\Delta(i)$ es un vector aleatorio generado con elementos dentro de un intervalo: [-1; 1].

Después de esto, cada bacteria i en su ciclo (j), en su ciclo de reproducción (k) y en su ciclo de eliminación-dispersión (l) modifica su posición como se indica en la ecuación 2.3.3.2.

Ecuación 2.3.3.2. [Passino, 2010]

$$\theta^i(j+1,k,l) = \theta^i(j,k,l) + C(i)\emptyset(i)$$

La ecuación 2.3.3.1 representa un giro (busca dirección) y la ecuación 2.3.3.2 representa un nado con tamaño de paso (C (i)). El nado será repetido Ns veces si la nueva posición es mejor que la anterior, es decir si las funciones de costo:

$$f\left(\theta^i(j+1,k,l)\right) < f\left(\theta^i(j,k,l)\right)$$

El paso de reproducción consiste en ordenar la población de bacterias i = 1,. . ., Sb basado en el valor de su función de costo $f\left(\theta^i(j,k,l)\right)$ y eliminar la mitad de aquellas con el peor valor Sr. La otra mitad será duplicada para mantener un tamaño de población constante.

El paso de eliminación-dispersión consiste en eliminar cada bacteria dentro de la población de forma aleatoria con probabilidad $0 \leq Ped \leq 1$.

Algoritmo 2.3.3.1. Pseudocódigo BFOA. [Passino, 2010]

Begin

Inicialización de parámetros

Generar una población inicial aleatoria

Evaluar $f\left(\theta^i(j,k,l)\right)$ para cada bacteria

For l=1 to Ned Do

For k=1 to Nre Do

For j=1 to Nc Do

For i=1 to Sb Do

Realizar el paso quimiotactico (giro-nado o giro-giro) para la bacteria i(j; k; l), los nados son limitados por Ns

End For

End For

Realizar el paso de reproducción para la eliminación de Sr bacterias y duplicar la otra mitad

End For

Realizar el paso de eliminación-dispersión para todas las bacterias i(j; k; l) 8i, i = 1; : : : ; Sb con probabilidad $0 \leq Ped \leq 1$

End For

End

Tabla 2.3.3.1. Parámetros utilizados por el BFOA. [Passino, 2010]

Nombre	Símbolo	Descripción
Bacterias	Sb	Numero de soluciones en la población (bacterias).
Paso quimiotactico	Nc	Número de veces que la población de bacterias podrá nadar o girar.
Nado	Ns	Número exacto en que la bacteria en procedimiento podrá nadar.
Reproducción	Nre	Número de veces que la población de bacterias se reproducirá.
Reproducir bacterias	Sr	Numero de bacterias que se reproducirán de toda la población.
Eliminación-dispersión	Ned	Número de veces en que se aplicara eliminación-dispersión en la población de bacterias.
Tamaño de paso	Ci	Es el paso que la bacteria en proceso podrá avanzar de un punto a otro en su búsqueda, limitado por las dimensiones del problema.
Probabilidad de eliminación-dispersión	Ped	Determina que bacterias de la población será eliminada y dispersada la nueva en el área de búsqueda.

2.3.4 Explicación del Pseudocódigo. Siguiendo los pasos mostrados en el pseudocódigo anterior, se describe el proceso que se realiza durante su ejecución con un problema de optimización.

- Como primer paso se inicializan los parámetros, numero de bacterias Sb, limite del paso quimiotáctico Nc, , limite del paso de nado Ns, el paso de reproducción Nre, numero de bacterias a reproducir Sr, limite del paso de eliminación-dispersión Ned, tamaño de paso Ci (depende de la dimensión del problema) y probabilidad de eliminación-dispersión Ped.

- Se procede a generar una población aleatoria de bacterias Sb, el tamaño de la población debe ser constante durante todo el proceso del algoritmo, el controlador inteligente trabaja con una población de 10 bacterias.

- Posteriormente se evalúan cada una de las bacterias Sb , es decir se sustituye en la función objetivo del problema a resolver los valores de las variables de la bacteria en proceso, esto se realiza para toda la población de bacterias.

- Se incrementa en 1 el ciclo de eliminacion-dispersion, reproducción y quimiotactico, Ned+1, Nre+1, Nc+1 hasta el número máximo establecido en la iniciación de parámetros.

- Posteriormente cada una de las bacterias realizan su primer ciclo quimiotactico, es decir, realizan su búsqueda nadando y avanzando hacia una nueva posición, los nados de cada una de las bacterias son solo el numero permitido por Ns, la elección de un nado se da cuando se mejoro el valor de la función objetivo en la nueva posición. De otra manera se realizara un giro.

- A continuación se realiza el proceso de reproducción, en el cual se reproducen la mitad de las mejores bacterias y se elimina a la otra mitad de las bacterias con peor posición dentro del espacio de nutrientes-

- Finalmente se realiza el proceso de eliminación-dispersión de bacterias de acuerdo a la probabilidad determinada por Ped.

- Se incrementan los ciclos, eliminación-dispersión, reproducción y quimio táctico y se vuelven a realizar lo descrito en los puntos 5, 6 y 7 hasta cumplir con el número máximo de ciclos establecidos en la iniciación de parámetros.

Como podemos observar los parámetros que utiliza el BFOA son numerosos y los ciclos que intervienen en el hacen que sea un algoritmo demasiado extenso, produciendo en primera instancia un número elevado de evaluaciones y por consiguiente un algoritmo lento. Además, la calibración de parámetros puede ser complicado para los usuarios. En la siguiente tabla se muestran los parámetros utilizados por el BFOA.

2.4. PLANTEAMIENTO DEL PROBLEMA

El control de sistemas una de las ramas de la ingeniería que mas participación tiene en la industria y en general en ramas modernas como la robótica, bioingeniería, etc.… Su impacto radica en que su estudio permite manipular sistemas para que operen de la forma deseada; el control puede ser aplicado para manipular elementos básicos como una válvula, un motor, e incluso sistemas complejos como órganos artificiales.

Para poder controlar un sistema se hace necesario básicamente:

- Estudiar su comportamiento y características para de esta forma obtener un modelo.

- Tener unas características deseadas de cómo queremos que se comporte el sistema controlado.

- Diseñar el controlador en base a las características deseadas y el modelo obtenido.

Sin embargo, para obtener el modelo del sistema generalmente se utilizan métodos off-line, esto quiere decir que se estiman los parámetros para modelar el sistema cuando este no se encuentra funcionando en un proceso real, la desventaja de esto es que algunos sistemas no se pueden desacoplar de un proceso para realizar su estudio y requieren de una estimación de parámetros online.

Para solucionar este inconveniente se han planteado varias teorías de identificación de sistemas online, algunas utilizando métodos matemáticos, y recientemente utilizando algoritmos de optimización bioinspirados.

Pensemos ahora en un sistema cualquiera al que se le han logrado estimar los parámetros que definen su comportamiento. ¿Qué pasaría si el comportamiento del sistema no es lineal en ciertos puntos?

Generalmente, para identificar un sistema se plantea un modelo lineal al cual se le calculan unos parámetros que incorporados al modelo definan mejor la respuesta del sistema que se desea identificar, esto plantea un problema ya que un sistema real no se comporta de manera lineal siempre. Se hace necesario crear un mecanismo de adaptación para que estos parámetros varíen dependiendo de la dinámica del sistema.

Como vemos, diseñar un controlador para un sistema es una tarea compleja si se cuentan con pocas herramientas y aun contando con herramientas computacionales para la obtención de ciertos datos hacen que todo el trabajo sea especial para el sistema que se desea controlar, no hay generalizaciones, y si se desea controlar otro sistema se tiene que repetir todo el proceso desde cero.

Por esta razón y a fin de evitar la parte más ardua del diseño de controladores, en este trabajo se plantea el desarrollo de un dispositivo: controlador inteligente, que sea capaz de controlar un sistema sin necesidad de realizar un modelo y simplemente definiendo una respuesta deseada del sistema. El dispositivo será entonces una plataforma inteligente capaz de generalizar el proceso para diseñar el controlador para cualquier sistema con tiempos de respuesta para los que el procesador del dispositivo pueda responder.

2.5. PROPUESTA DE IMPLEMENTACIÓN

Para implementar una solución al problema planteado anteriormente debemos recordar los conceptos del capítulo uno, para generar una propuesta uniendo todas las definiciones. En general hablamos del sistema operativo FreeRTOS, de las Colas o Queues como estructuras de datos, de los circuitos de señal mixta PSoC y de un esquema general de control adaptativo.

La plataforma base sobre la que se implementara todo el sistema de control inteligente serán los circuitos de señal mixta PSoC 5 que internamente poseen un procesador ARM Cortex M3 a 80Mhz, una velocidad aceptable para realizar los cálculos necesarios para implementar los algoritmos de optimización por enjambre de bacterias y control adaptativo. Sin embargo, es optimo implementar todo el sistema dentro del PSoC como tareas independientes que den la ilusión de procesarse en paralelo, esto debido a que si se implementará todo el sistema como un proceso secuencial el sistema podría no responder en tiempo real y el objetivo de controlar un sistema no se podría cumplir.

El programa de control inteligente implementado sobre el PSoC 5 se dividirá en 7 tareas, cada una es un bloque dentro del esquema de control adaptativo, solamente una no está definida dentro de este tipo de esquema – la tarea “Display” – y sirve para obtener información visual de variables internas del controlador inteligente. Las tareas se han nombrado así:

- Ref – (Referencia)
- Sum – (Sumador)
- PID – (Controlador PID)
- System – (Sistema)

- Ident – (Identificador)
- Tuning – (Ajuste de parámetros del PID)
- Display – (LCD de caracteres)

La función de cada una estas tareas es explicada en la siguiente sección junto con un diagrama donde se puede percibir la relación que tienen cada una de estas tareas y como juntas producen la respuesta deseada para el controlador inteligente.

Para lograr esta separación por tareas y una ilusión de concurrencia es necesario utilizar un sistema que defina un tiempo de ejecución para cada tarea y cuando acabe su tiempo de ejecución programe la ejecución de otra tarea y así con todas las tareas, esta función la proporciona un sistema operativo en tiempo real, en este caso FreeRTOS.

FreeRTOS será el encargado de administrar los tiempos de ejecución de cada tarea y evitar posibles bloqueos que puedan perturbar la función elemental del controlador inteligente, es importante que sea así ya que si la ejecución de los algoritmos dentro del PSoC 5 fuera secuencial muy posiblemente algunas tareas como “Ident” que consume bastante tiempo de ejecución en el procesador podrían provocar un colapso del sistema.

Cada tarea realiza un procesamiento de datos y lo envía a alguna otra tarea o periférico, la mayoría de las tareas que de ahora en adelante llamaremos bloques tienen entradas y salidas, algunos solamente salidas, estos bloques son similares a los bloques que se dibujan en los diagramas de bloques en los cursos de control.

Las señales o interconexiones entre bloques se implementan en el controlador inteligente como forma de comunicación entre estos, podría pensarse en un principio en ver estas señales como variables dentro del algoritmo pero esto presenta muchos inconvenientes, uno de ellos es el acceso a estas variables. Supongamos por un momento que un bloque realiza un procesamiento y la salida de este bloque es enviado a otro mediante una variable, si ambos bloques están funcionando en paralelo habrían problemas cuando un bloque quiera leer esta

variable mientras el otro trata de leerla, posiblemente habrían perdidas de datos ya que puede darse el caso que un bloque escriba esta variable varias veces mucho antes que el otro bloque pueda leerla.

La solución a este problema se llama Colas o Queues, estas estructuras ya vienen dadas por FreeRTOS y se define una Cola por cada una de las señales que comunican cada uno de los bloques. Dado el caso que un bloque como “Ident” no entregue los datos de salida a los otros bloques en el plazo que tiene para hacerlo, la solución es procesar el último dato que se proceso en la cola para así no perjudicar los plazos que tienen los otros bloques para realizar el procesamiento.

La arquitectura de PSoC 5 nos brinda aun más beneficios, al tener bloques análogos y digitales podemos liberar carga del procesador, este es el caso de la generación de números aleatorios necesarios para la ejecución del algoritmo de optimización por enjambre de bacterias. Los bloques digitales del PSoC serán utilizados para crear una secuencia de números pseudo-aleatorios con arreglos de compuertas XOR, de esta forma el algoritmo de optimización puede ejecutarse mucho más rápido.

El bloque “Ident” también requiere obtener muestras de la entrada y salidas del sistema para su posterior procesamiento, podemos liberar el procesador de la adquisición de estos datos utilizando un componente que posee PSoC 5 que copia de datos desde un periférico hasta la memoria o viceversa sin intervención del procesador, este componente es llamado DMA (Direct Memory Access).

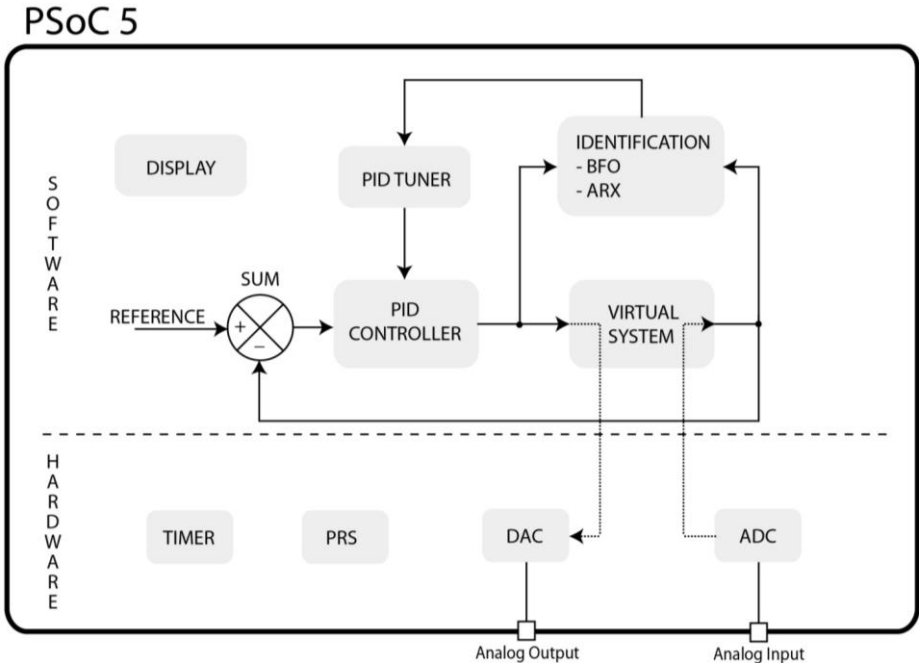
Una vez implementados todos los algoritmos necesarios para el controlador inteligente podemos probarlo definiendo un sistema como ecuación en diferencias dentro del bloque “System” o adquiriendo y enviando señales a un sistema real utilizando el mismo bloque.

2.6. DIAGRAMA DE IMPLEMENTACIÓN

Una forma de entender mejor la propuesta de implementación planteada es mediante el uso de diagramas, en esta sección se dibuja un diagrama que define cada uno de los componentes de controlador inteligente y la interconexión entre ellos.

La siguiente figura es el diagrama de implementación, en la figura se definen los componentes software y hardware necesarios, los componentes software son en realidad cada una de las tareas ejecutadas y administradas por el sistema operativo en tiempo real FreeRTOS, algunas tareas dependen de datos obtenidos desde componentes hardware, otras solamente realizan procesamiento de datos, en la figura los cuadros con etiquetas “Analog Output” y “Analog Input” representan conexiones con el mundo físico, es decir son pines del circuito de señal mixta PSoC 5.

Figura 2.1 Diagrama simplificado del controlador inteligente.



2.6.1. Timer. Hemos hablado del uso de un sistema operativo en tiempo real, pero ¿Cómo FreeRTOS sabe que debe cumplir con los plazos estipulados para un sistema en tiempo real?, la respuesta es que usa un timer, el timer define el tiempo en el que FreeRTOS ejecuta sus algoritmos de planificación de tareas para administrar los tiempo de ejecución de cada tarea, en la implementación del controlador inteligente se definió que esto se realiza cada 1ms, tiempos más grandes perjudicarían la noción de tiempo real del sistema y tiempos más cortos harían que FreeRTOS ocupe más tiempo de ejecución que las tareas y por tanto el sistema como tal seria ineficiente.

2.6.2. PRS. PRS (Pseudo Random Sequence) en español Secuencia Pseudo Aleatoria, es el bloque encargado de generar números aleatorios para su uso en los algoritmos de optimización, este bloque es completamente independiente del procesador, lo que quiere decir que este bloque genera números pseudo-aleatorios mientras el procesador puede estar ejecutando algoritmos, de esta forma logramos evitar el consumo del procesador y hacer más eficiente el sistema.

2.6.3. ADC. Es el componente hardware que nos permite tomar datos desde el exterior del PSoC, mediante este conversor análogico-digital obtenemos datos de la salida del sistema físico que se desea controlar, el tiempo de muestreo se define según los criterios de nyquist que hacen que se pueda controlar el sistema de la forma deseada y la resolución de la precisión que con que se desea medir la respuesta del sistema.

2.6.4. DAC. Este bloque es un conversor digital-análogico, gracias a este componente hardware podemos enviar información al exterior del PSoC mediante señales de voltaje que llegaran a la entrada del sistema que se desea controlar. En el PSoC este conversor está limitado a una resolución de 8bits y tiempo de respuesta de 4mbps, es posible utilizar también un conversor digital-análogico externo.

2.6.5. Reference. En realidad este bloque no realiza un procesamiento de datos como tal, en cambio define la respuesta que esperamos del sistema, es manipulado por el operador del controlador inteligente que puede ser humano o algún dispositivo electrónico.

2.6.6. Sum. Es el bloque de procesamiento más liviano su única función es realizar una resta de la referencia y la salida del sistema para así obtener un error.

2.6.7. Virtual System. Una de las ventajas del controlador inteligente es que no es necesario tener siempre un sistema físico para realizar el control, este bloque puede conectarse a un sistema físico mediante los ADC y DAC o en el puede definirse un sistema en ecuación en diferencias para realizar simulación. En resumen este bloque es el sistema como tal que se desea controlar, puede ser un sistema físico o un sistema digital.

2.6.8. Identification. Es el bloque con más carga de procesamiento, por esta razón y para no perturbar los plazos de procesamiento de otros bloques el controlador inteligente hace que el asincronismo para esta tarea sea posible. En este bloque se ejecutan los algoritmos de optimización por enjambre de bacterias y algoritmos ARX para la obtención de un modelo de la planta con datos obtenidos de muestra hechas a las entradas y salidas. Este bloque permite configurar un modelo diferente al ARX, en este proyecto se utiliza el modelo ARX ya que es el más sencillo y no posee muchos términos para calcular. El algoritmo de optimización por enjambre de bacterias se encuentra configurado por defecto con una población de 10 bacterias aunque este valor puede ser modificado, el espacio de búsqueda tiene dos dimensiones ya que deseamos buscar los parámetros A y B del modelo ARX para un sistema de primer orden, si se desea calcular los parámetros para un sistema de segundo orden el usuario debe implementar las funciones de costo y ajuste de parámetros del PID y cargarlas como punteros dentro del programa, se ha definido la cantidad de pasos chemotacticos, eventos de reproducción, eliminación y dispersión como 4, el largo de los pasos chemotacticos será de 0.1 y máximo se hacen 4 veces estos pasos. Se eligieron los parámetros anteriores para no saturar al procesador del dispositivo pero en la misma medida evitar que el algoritmo pueda volverse ineficiente.

La función de costo para el algoritmo de optimización es el cálculo del error del modelo ARX propuesto por cada bacteria contra el modelo real. Es decir para cada instante se calcula la siguiente función que estima una salida actual con las muestras pasadas y parámetros a y b de cada bacteria:

Ecuación 2.6.8.1. Estimación de la salida actual. [Gomez, 2011]

$$\hat{y} = -ax[n-1] + by[n-1]$$

Ahora para obtener el error se calcula:

Ecuación 2.6.8.2. Error cuadrático medio. [Gomez, 2011]

$$error = \frac{1}{N}\sum_{k=1}^{N} \|y[k] - \hat{y}[k]\|^2$$

Donde N es el total de muestras, $y[k]$ la salida actual y $\hat{y}[k]$ la salida estimada del sistema.

Los anteriores cálculos se realizan para cada bacteria dentro del algoritmo y para todas las iteraciones en las cuales se trata de buscar la bacteria con menor error posible. Cuando se termina de ejecutar el algoritmo de optimización y se registran los datos de salida, estos se toman como datos de entrada para volver a ejecutar el algoritmo y así lograr una mejor estimación del modelo de la planta, esto se hace en un ciclo que empieza desde que se enciende el controlador inteligente hasta que se apaga.

La salida de este bloque son los parámetros a y b que mejor modelan la planta real.

2.6.9. PID Tuner. Una vez identificado el sistema este bloque se encarga de hacer la transformación de la información obtenida del bloque de identificación a constantes Ki y Kp necesarias para el funcionamiento del controlador PID.
Del bloque de identificación obtenemos a y b y los sustituimos en las siguientes ecuaciones para obtener Ki y Kp:

Ecuación 2.6.9.1. [Ogata, 2006]

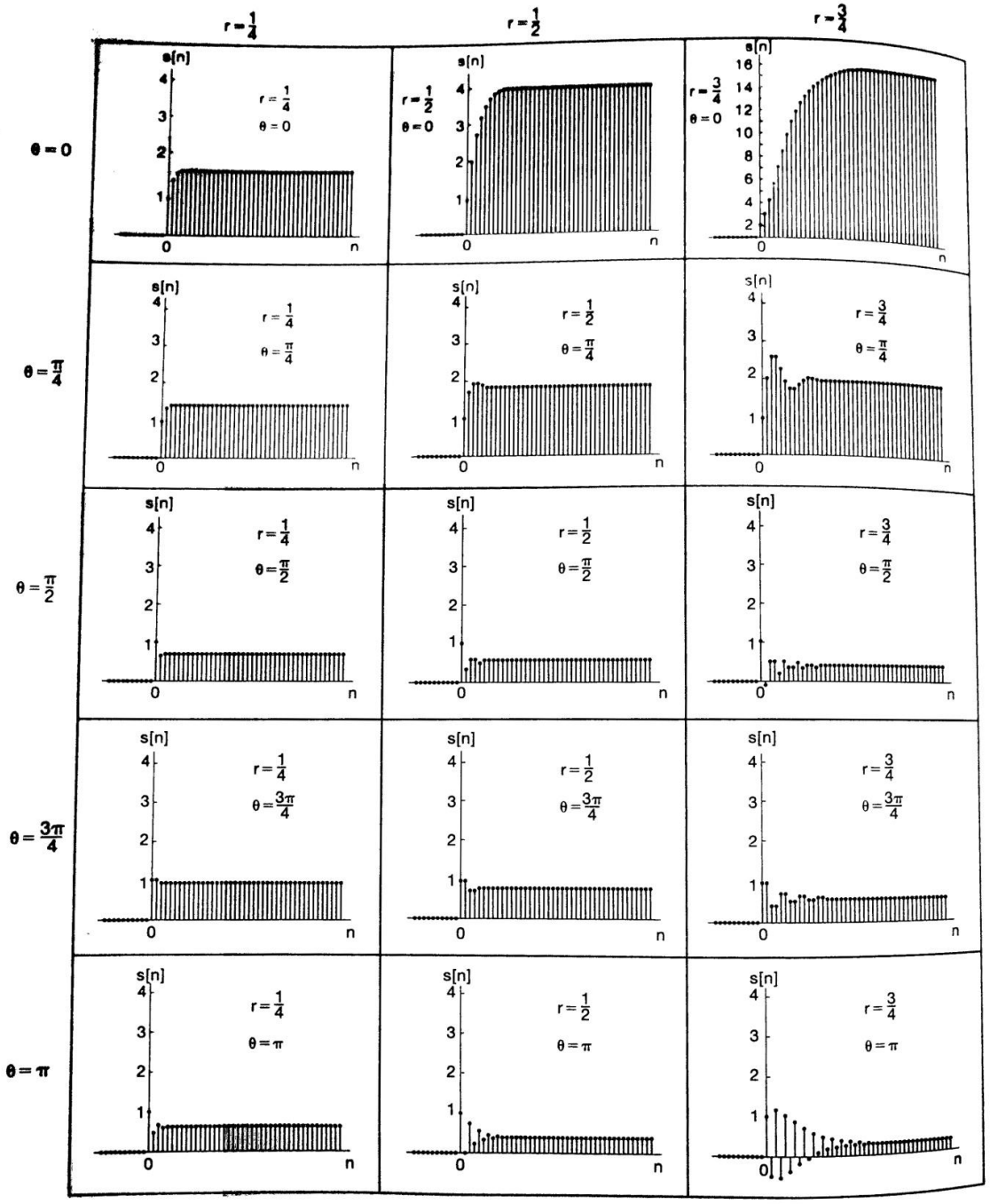
$$Kp = \frac{a - r^2}{b}$$

Ecuación 2.6.9.2. [Ogata, 2006]

$$Ki = \frac{r^2 + (1 - r\cos\theta)}{b}$$

Recordemos que al controlador inteligente se le deben asignar unos parámetros de la respuesta deseada para el sistema en lazo cerrado, estos parámetros son r y theta que son simplemente la ubicación de un polo dentro del círculo unitario en Z. El operador del controlador inteligente puede asignar este polo tomando los datos de una tabla que muestra la relación de r y theta y la respuesta que tendrá el sistema para estos valores, la tabla es la siguiente:

Tabla 2.1 Respuesta de un sistema a un escalón con parámetros r y theta variables.



2.6.10. PID Controller. La implementación discreta de un controlador PID se encuentra en este bloque, las constantes Ki Kp y Kd son en realidad variables y dependen de las señales de salida del bloque “PID Tuner”.

Para calcular un controlador PID en tiempo discreto tenemos que realizar las siguientes operaciones:

Ecuación 2.6.10.1. Diferencia entre respuesta actual y respuesta deseada. [Ogata, 2006]

$$e(n) = R - y(n)$$

Ecuación 2.6.10.2. Calculo de una integral en tiempo discreto. [Ogata, 2006]

$$I(n) = 2I(n-1) + \frac{\big(e(n) - e(n-1)\big)T}{2}$$

Ecuación 2.6.10.3. Calculo de una derivada en tiempo discreto. [Ogata, 2006]

$$D(n) = \frac{e(n) - e(n-1)}{T}$$

Ecuación 2.6.10.4. Calculo de un PID discreto. [Ogata, 2006]

$$PID(n) = K_p e(n) + K_i I(n) + K_d D(n)$$

Primero obtener un error $e(\ \ )$ que esta dado por la resta entre la referencia y la salida actual de la planta.

Luego hallar la integral del error, esto se hace calculando I(n), y hallar la derivada del error calculando D(n), en ambas ecuaciones T es el tiempo de muestreo.

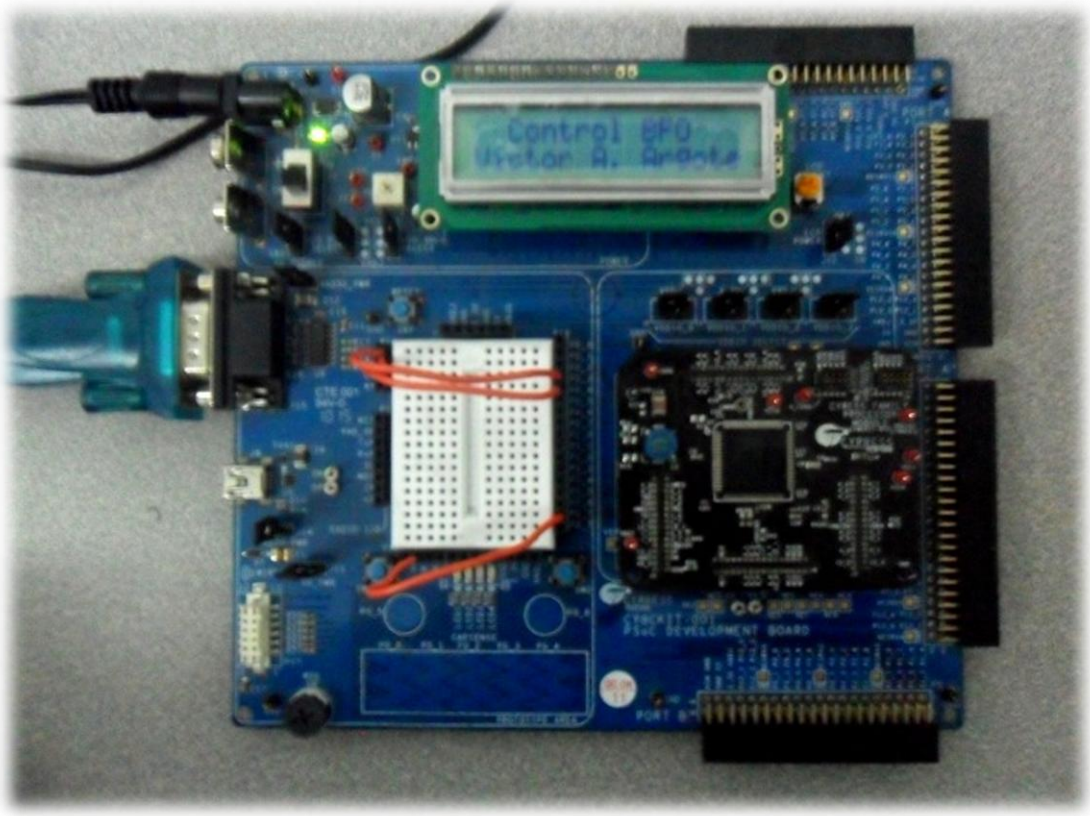
La salida del controlador PID será entonces PID(n) para ese instante de tiempo.

2.7. PLATAFORMA DE IMPLEMENTACIÓN

El soporte hardware en donde estarán programados todos los algoritmos que permiten al controlador inteligente realizar su tarea esta dentro del núcleo del PSoC 5, en un procesador ARM Cortex M3 corriendo a 80Mhz y algunos bloques digitales y análogos del PSoC 5.

El circuito de señal mixta PSoC 5 es fabricado por Cypress Semiconductor, fue lanzado al mercado en el 2009 y hasta la fecha solo Cypress Semiconductor ofrece un kit de desarrollo para toda la familia de PSoCs. El kit es llamado CY8CKIT-001 y lo podemos apreciar en la siguiente imagen:

Figura 2.2 Kit de Desarrollo CY8CKIT-001



El kit de desarrollo CY8CKIT-001 provee acceso a todos los pines del PSoC, cuenta con puertos de expansión, posee un slot para conectar pantallas LCD, se puede realizar comunicación a él mediante UART, Wireless o puerto USB, también se tiene acceso en la misma tarjeta a la tecnología Capsense.

Capsense es una tecnología de sensores táctiles en PCB desarrollados por Cypress Semiconductor, el kit de desarrollo CY8CKIT-001 cuenta con dos botones y un slider capsense.

Al momento de realizar este trabajo Cypress Semiconductor aun no vende el chip del circuito de señal mixta PSoC 5, lo que ha imposibilitado la realización de una tarjeta PCB para el propósito específico de este trabajo. Aun así el kit de desarrollo de Cypress Semiconductor posee todas las herramientas necesarias para cumplir los objetivos del trabajo.

El controlador inteligente utilizará el ADC y el DAC dentro del chip PSoC para realizar la medición de la salida de la planta que se desea controlar y envío de señal a su entrada.

El ADC puede funcionar a 16bits de resolución, mientras que el DAC solo puede funcionar a 8bits, esta es solo una limitante en la implementación, y en la puesta en marcha del controlador se ha comprobado que ejecuta muy bien su función aun con esta limitante.

El sensor y actuador conectado al controlador inteligente debe tener una configuración de 0-5V.

Del kit de desarrollo también utilizaremos el LCD incorporado para mostrar información sobre el controlador inteligente, principalmente medición de las señales de control, referencia o de entrada y salida de la planta.

Con los algoritmos y herramientas ya implementadas sobre la plataforma PSoC 5 ya podemos realizar un análisis del comportamiento del controlador inteligente con lo que se abre un espacio para la realización de pruebas y resultados.

3. RESULTADOS EXPERIMENTALES

Es esencial ahora que ya está definido e implementado el controlador inteligente realizar las pruebas necesarias para validar el correcto funcionamiento del controlador y evaluar los resultados para caracterizar áreas de aplicación de esta plataforma, comprender las limitantes y también ventajas de usar un controlador inteligente para la automatización de distintos procesos.

3.1. DESCRIPCIÓN DE LA PRUEBA

El controlador inteligente puede realizar automáticamente dos procesos que generalmente al diseñar un controlador son esenciales.
Estos dos procesos son:

- Identificar los parámetros del sistema que se desea controlar.
- Diseñar un controlador PI a partir de los parámetros del sistema identificado.

Ambos procesos se realizan en “paralelo” gracias al sistema operativo en tiempo real FreeRTOS incorporado en el controlador inteligente.

Ahora deseamos saber las ventajas y desventajas de utilizar el controlador inteligente. En este capítulo se incorpora el controlador inteligente a una planta de presión del laboratorio de automatización de la Universidad Autónoma de Occidente, esto para comparar los datos obtenidos mediante el controlador inteligente con los datos obtenidos de un controlador por observador de estados para la misma planta.

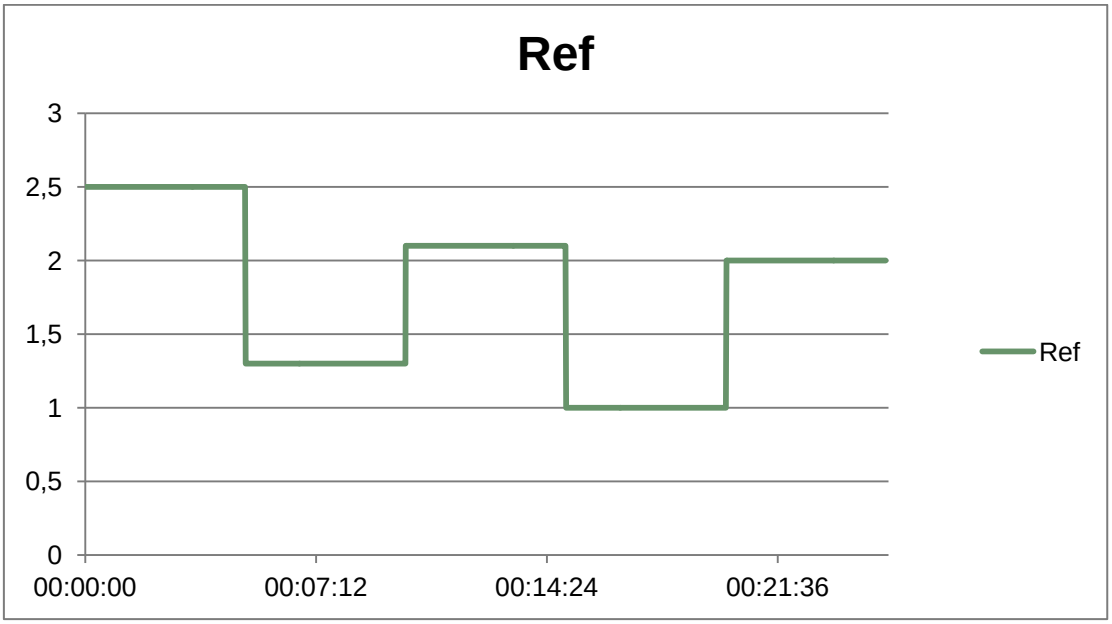
La comparación se realiza con un controlador por observador de estados ya que este tipo de controlador realiza estimaciones de la salida de la planta, función que no cumple otro tipo de controladores; de esta forma podemos comparar una variable fundamental en el rendimiento del controlador inteligente. El diseño del controlador por observador de estados no se describirá aquí puesto que no es tema de este trabajo.

Se realizaran dos pruebas, en la primera se acopla el controlador inteligente o el controlador por observador a la planta y se toman datos de todas las señales y la

segunda donde se acoplan de nuevo y se toman datos pero esta vez se suman perturbaciones al sistema para ver la respuesta de ambos controladores. Ambas pruebas son realizadas bajos las mismas condiciones, la idea es enviarle a los controladores las mismas entradas de referencia durante el mismo periodo de tiempo y analizar todas las señales que se obtienen. La señal de referencia que se le envía a ambos controladores son escalones espaciados por un tiempo de 5 minutos entre cada uno, los niveles de voltaje de cada escalón son sucesivamente 2,5V, 1,3V, 2,1V, 1V y 2V.

En la siguiente grafica podemos apreciar los escalones que son tomados como referencia para obtener los datos de comportamiento de cada controlador frente a estas entradas:

Figura 3.1 Señal de referencia para las pruebas.



Luego de someter ambos controladores a la misma referencia se toman datos de la señal de salida de la planta y la salida del controlador. Con estos datos procedemos a calcular lo siguiente:

Índices de desempeño de error: Evalúan la diferencia entre el valor deseado $r(k)$ y el valor obtenido en la salida $y(k)$. Se definen dos índices:

Ecuación 3.1.1. Promedio del Error al Cuadrado [Marulanda, 2007]

$$J_1 = \frac{1}{N}\sum_{k=0}^{N}\big(r(k) - y(k)\big)^2$$

Ecuación 3.1.2. Promedio del Error Absoluto [Marulanda, 2007]

$$J_2 = \frac{1}{N}\sum_{k=0}^{N}|r(k) - y(k)|$$

Índice de esfuerzo de control: Evalúa el promedio del valor absoluto de la acción de control $u(k)$.

Ecuación 3.1.3. [Marulanda, 2007]

$$J_3 = \frac{1}{N}\sum_{k=0}^{N}|u(k)|$$

Índice de suavidad en el esfuerzo de control: Compara la diferencia entre una muestra anterior de la señal de control $u(k-1)$ y la actual $u(k)$. Permite conocer que tan bruscos son los cambios de la señal de control.

Ecuación 3.1.4. [Marulanda, 2007]

$$J_4 = \frac{1}{N-1}\sum_{k=1}^{N}\big(u(k) - u(k-1)\big)^2$$

Índices de seguimiento: Evalúa la diferencia entre la salida real del sistema $y(k)$ y la salida estimada por el controlador $\hat{y}(k)$.

Ecuación 3.1.5. [Marulanda, 2007]

$$J_5 = \frac{1}{N}\sum_{k=0}^{N}\big(y(k) - \hat{y}(k)\big)^2$$

En la última sección de este capítulo se presentarán los resultados en graficas junto con una tabla comparativa para resaltar elementos claves del desempeño de cada controlador.

3.2. SISTEMA PROPUESTO PARA LA PRUEBA

Para realizar la prueba del controlador inteligente se utilizará la planta de presión que se encuentra ubicada en el laboratorio de automática de la Universidad Autónoma de Occidente, los datos se registran directamente con el controlador inteligente que es capaz de tomar todas las señales y enviarlas en formato ASCII a un computador a través de un puerto USB.

El sistema propuesto cuenta con una entrada de aire que va conectada a una válvula electrónica y de esta a dos tanques metálicos con válvulas de escape, la válvula electrónica tiene una entrada de voltaje (0-5V) para controlar el actuador que se encarga de regular el paso de aire y conectado a los tanques metálicos un sensor que tiene una salida de voltaje (0-5V) para leer los datos que el sensor de presión entrega.

En la siguiente figura podemos ver la planta de presión junto con el controlador inteligente que se encarga de ejecutar el control de los niveles de presión de los tanques metálicos:

Figura 3.2 Planta de pruebas.

En la parte inferior izquierda podemos apreciar los tanques metálicos a los que se les llenará con aire para generar la presión deseada, sobre la parte superior del primer tanque (arriba) se encuentra ubicado el sensor de presión, ambos tanques se conectan mediante una manguera y tienen una válvula de escape, esta se abre un poco girándola 90° para la prueba propuesta, siendo 0° su posición cuando está totalmente cerrada, la válvula se encuentra totalmente abierta cuando se gira 1800°. El elemento azul que se encuentra en el centro de la imagen es la válvula electrónica que se encarga de regular la entrada de aire a los tanques. En la parte inferior derecha sobre la mesa blanca se encuentra el kit de desarrollo PSoC5 que controlando la planta y conectado al tablero en la parte superior derecho, dicho tablero tiene las borneras que permiten enviar niveles de voltajes al actuador y recibir niveles de voltaje desde el sensor.

3.3. ANÁLISIS DE SISTEMA PROPUESTO

Después de poner en funcionamiento la planta de presión se puede deducir ciertas características del sistema que son necesarias conocer para evitar dar al controlador inteligente parámetros incorrectos o señales de referencia que por razones intrínsecas del sistema no se pueden cumplir.

Una de las características de la planta de presión es que no puede registrar más de 80psi de presión, esto debido al sensor como tal. Los 80psi de presión registran 3 voltios a la salida del sensor y un voltaje mínimo de 1V cuando los tanques se encuentran a 0psi, es decir que la referencia que se ponga en el controlador no puede sobrepasar los 3 voltios ni ser inferior a 1V en caso contrario la planta de presión no responderá nunca a la referencia definida en el controlador.

Las válvulas de escape de los tanques metálicos tienen importancia en el comportamiento del controlador y en la función de transferencia de la planta, si se encuentran muy abiertos los valores de presión máxima disminuyen y si se encuentran completamente cerradas sería imposible bajar de un nivel de presión alto a uno bajo es por esto que las válvulas de escape de los tanque metálicos se encuentran abiertas solo un poco, concretamente 90° desde la posición en que se encuentran totalmente cerrados.

También es necesario destacar el uso necesario de un filtro a la salida del sensor para reducir el ruido en las mediciones y mejorar la respuesta del controlador.

Luego de haber realizado el análisis al sistema y teniendo en cuenta todas estas variables ya podemos proceder a conectar el controlador y configurarlo de manera correcta para que opere.

3.4. COMPORTAMIENTO DESEADO PARA EL SISTEMA

El controlador inteligente necesita que se definan unos parámetros para llevar al sistema a un comportamiento deseado, este comportamiento define básicamente el tiempo que puede tardar el sistema en llegar a la señal de referencia y estabilizarse. En este caso los parámetros para el comportamiento deseado son dos variables θ y r que definen la ecuación característica de una función de transferencia en Z, esta ecuación característica la que define el tiempo de estabilización para el controlador.

Ecuación 3.4.1. Ecuación característica en Z [Oppenheim, 1998]

$$z^2 + 2rz\cos\theta + r^2$$

Ecuación 3.4.2. Ecuación característica en S [Oppenheim, 1998]

$$S^2 + 2\gamma\omega_n S + {\omega_n}^2$$

La planta de presión es un sistema de respuesta lenta, esto por razones físicas que hacen imposible los cambios bruscos en los niveles de presión registrados por el sensor, tal motivo debe tenerse en cuenta para poder definir un comportamiento deseado de la planta, en tiempo continuo el tiempo de estabilización y la constante γ definen este comportamiento, debido a que el controlador inteligente utiliza variables en tiempo discreto debemos hacer la conversión de la ecuación 3.4.2 a la ecuación 3.4.1.

Para realizar la prueba se ha definido un tiempo de estabilización de 120s y la constante γ igual a 1, se eligieron estos datos debido a lo mencionado anteriormente, para estos datos obtenemos en tiempo discreto:

$$r = 1$$
$$\theta = 0$$

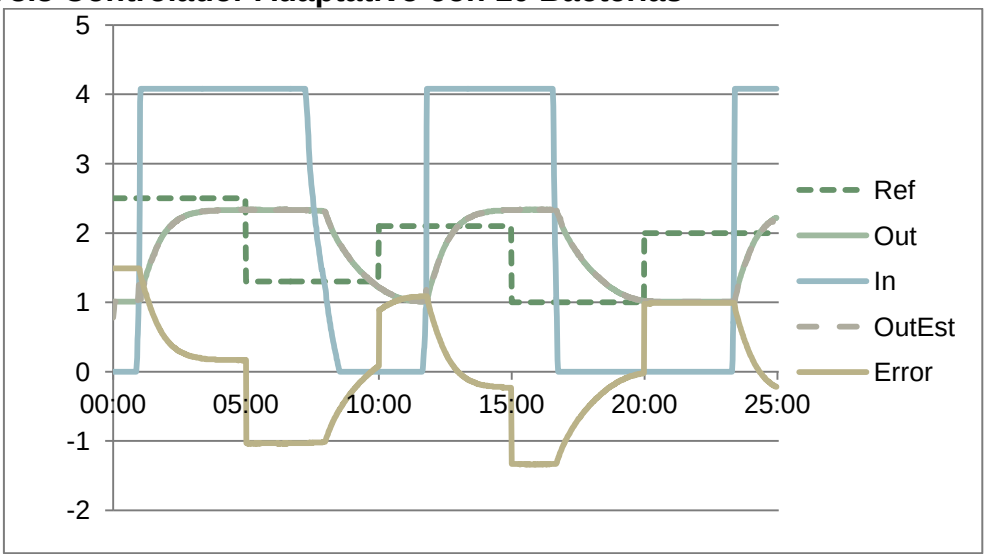
Con estos datos cargados en el controlador inteligente se define el comportamiento deseado para el sistema.

3.5. DATOS OBTENIDOS

Se realizaron cuatro pruebas en total, dos para cada tipo de controlador, se probó el comportamiento de cada uno frente a el sistema en estado normal y con perturbaciones, la perturbación que se realizó para la segunda prueba fue modificar el estado de las válvulas de escape de la siguiente forma: La prueba dura en su totalidad 25 minutos, al momento de iniciar las válvulas de escape están cerradas, a los 5 minutos se abren completamente, a los 8 minutos se cierran de nuevo, al minuto 11 se gira la válvula 90° para abrirla un poco y en el minuto 17 se cierra de nuevo la válvula por completo.

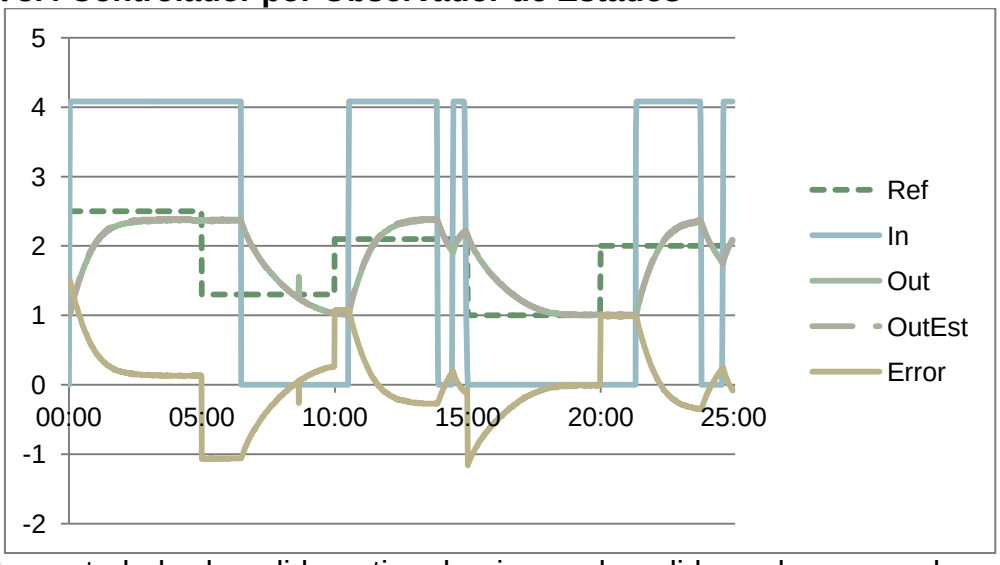
La figura 3.5.1 muestra las señales registradas desde el controlador adaptativo funcionando con 10 bacterías para la identificación, la figura 3.5.2 muestra el comportamiento del controlador por observador de estados, la figura 3.5.3 el controlador adaptativo funcionando con 16 bacterias y con las perturbaciones descritas anteriormente – se eligieron 16 bacterias para mejorar el modelo obtenido por el algoritmo, de igual forma se registra en la figura 3.5.4 el controlador por observador de estados funcionando con estas perturbaciones.
Al final podemos apreciar todos los índices de desempeño en una tabla, con estos datos podemos realizar conclusiones y realimentarnos con estos datos para mejorar el controlador utilizando algoritmos de optimización.

Figura 3.3 Controlador Adaptativo con 10 Bacterias



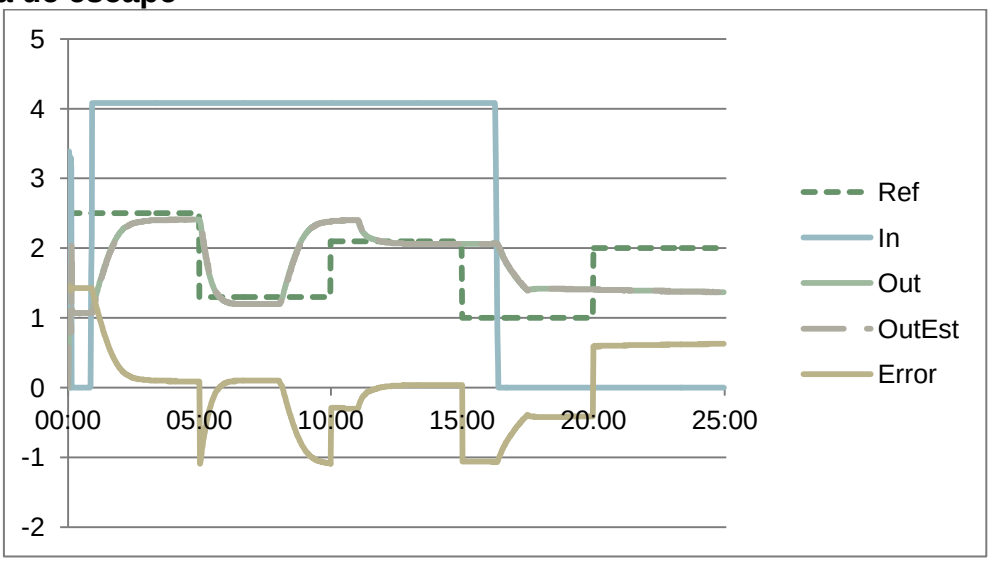
En el controlador adaptativo la salida estimada sigue a la salida real pero responde muy lento a los cambios en la referencia.

Figura 3.4 Controlador por Observador de Estados



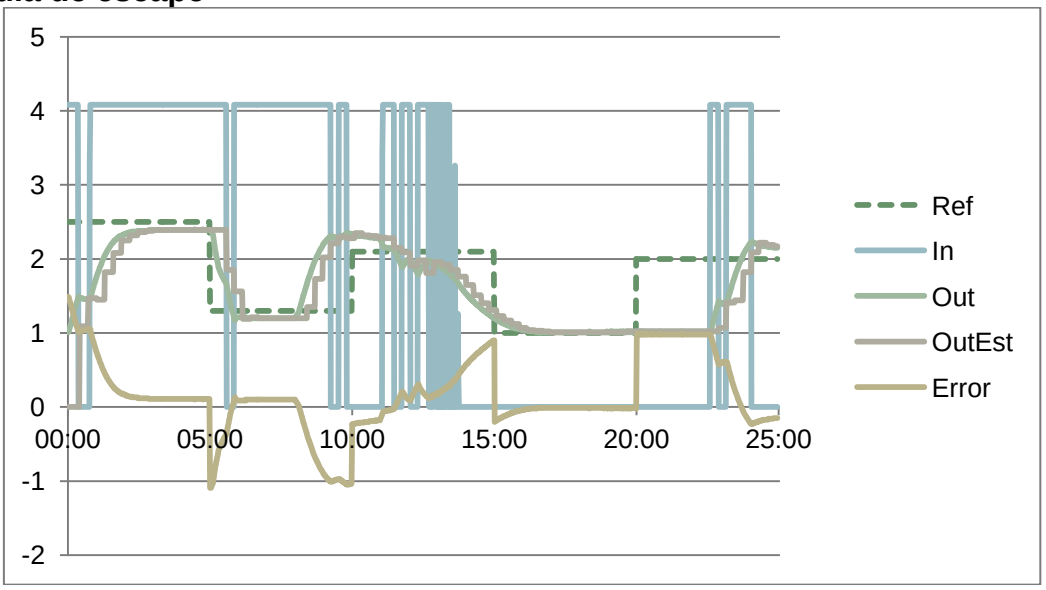
En este controlador la salida estimada sigue a la salida real y responde un poco más rápido a los cambios en la referencia.

Figura 3.5 Controlador Adaptativo con 16 Bacterias y perturbaciones en la válvula de escape



Aquí el controlador adaptativo trata de seguir la referencia aunque la entrada del sistema no varía mucho. La salida estimada sigue siempre a la salida real del sistema.

Figura 3.6 Controlador por Observador de Estados con perturbaciones en la válvula de escape



Las variaciones en la entrada del sistema utilizando este controlador son muchas y no sigue muy bien la referencia, la salida estimada tampoco es sigue a la salida real.

Cuadro 3.1 Indices de desempeño

	Prueba Normal		Prueba con Perturbaciones	
Controlador	Adaptativo	Observador de Estados	Adaptativo	Observador de Estados
J1	0,671266044	0,279239026	0,352780387	0,270087191
J2	0,682428286	0,381527685	0,450126751	0,357858572
J3	2,159052702	2,142961975	2,528005337	1,886170781
J4	0,011412275	0,054298065	0,03402535	0,421094463
J5	0,00032555	0,000338826	0,000932488	0,056827885

Los datos obtenidos indican que es necesario realizar más pruebas y mejoras en el algoritmo, la tabla comparativa entre los índices de desempeño de los controladores probados en este trabajo muestran buenos datos en cuando a la variabilidad de la señal de control que es muy baja para el controlador con algoritmos de optimización, las graficas muestran que el controlador por observador de estados no responde muy bien ante perturbaciones externas debido a que la mayor parte del tiempo la salida estimada por el controlador por observador de estados no sigue a la salida real de la planta, esta es una limitación que no tiene el controlador realizado en este proyecto, incluso el índice J5 que mide que tan bien el controlador estima la salida real del sistema es muy baja para el controlador adaptativo, aunque prácticamente igual al del controlador por observador de estados para la planta sin perturbaciones.

El error entre la referencia y la salida de la planta sin embargo no es la mejor para el controlador adaptativo, es siempre alto en todas las pruebas pero se puede mejorar realizando unas modificaciones en el algoritmo, básicamente las bacterias que optimizan la estimación del modelo de la planta se basan en un error cuadrático que tiene en cuenta la diferencia entre la referencia y la salida real de la planta, sabemos que si un valor menor que 1 se eleva al cuadrado este se vuelve incluso más pequeño de forma que calculando el valor absoluto del error en vez del error al cuadrado es muy probable que se pueda lograr un mejor comportamiento. Con esta modificación aceleramos el tiempo de ejecución y mejoramos la respuesta obtenida por el controlador para este caso sin cambiar los fundamentos teóricos del algoritmo.

4. CONCLUSIONES Y TRABAJO FUTURO

En este capítulo se describen las conclusiones obtenidas después de investigar, experimentar y evaluar el rendimiento del controlador inteligente con algoritmos de optimización basados en bacterias, asimismo se describen posibles trabajos futuros y mejoras al sistema.

4.1. CONCLUSIONES

Realizar una implementación de un esquema de control adaptativo con algoritmos bioinspirados sobre un circuito de señal mixta PSoC trae consigo mejoras respecto a la forma en cómo se concibe el diseño de controladores. Vemos que con el controlador inteligente implementado en este trabajo no se hace necesario el uso de programas externos o cálculos para la realización del controlador, es necesario solamente definir parámetros deseados para un sistema y con base a estos datos el dispositivo se hace cargo de la parte más tediosa del diseño de controladores.

La única limitante es el tiempo de respuesta del sistema, ya que si este es muy pequeño se hace necesario un procesador más veloz para que el bloque de identificación logre entregar un dato en el tiempo requerido por el sistema. Utilizar el sistema operativo FreeRTOS fue de gran ayuda y minimizo tiempo valioso en el desarrollo, se logro incluso implementar transmisión de datos por USB para ambos controladores, en el momento de la toma de datos solo fue necesario abrir una plantilla en Excel con los gráficos haciendo referencia a celdas especificas y conectar el PSoC 5 por USB, este fue detectado por el sistema operativo Windows como un teclado y todos los datos ingresaban a Excel como si manualmente se estuvieran escribiendo.

Las ventajas de la plataforma PSoC son enormes, el time-to-market de cualquier producto se reduce gracias al concepto de reutilización de componentes, la lista de componentes electrónicos disminuye lo que disminuye también el tamaño de la PCB que se debe diseñar para un producto basado en PSoC, lo único lamentable al momento de realizar esta tesis es que el chip PSoC 5 no se encuentra disponible en el mercado haciendo imposible la implementación de un hardware completo.

Los controladores basados en algoritmos de optimización a diferencia de los controladores basados en operaciones matemáticas directas, requieren mucho

tiempo de procesamiento siendo este un factor importante para la eficiencia del controlador ya que entre mayor muestras se tenga la estimación de parámetros va a ser más precisa pero el tiempo requerido de procesamiento será mayor y de no tener una plataforma potente estos cálculos no se podrían entregar los resultados en tiempo real, fue por este motivo que en este trabajo a cada bloque dentro del controlador se le implemento como una tarea independiente y trabajando en paralelo con otras para evitar los cuellos de botella posiblemente causados por la lentitud de las tareas de identificación. Al final de todo el trabajo realizado se concluye que el control adaptativo basado en algoritmos de optimización para la identificación de la planta es una buena solución para el desarrollo de controladores que requieran tiempo de respuesta no muy alto, aunque también es cierto que quedan técnicas por explorar como la lógica difusa y el control predictivo, es probable que se obtengan resultados muy satisfactorios de la fusión de estas técnicas.

4.2. TRABAJO FUTURO Y MEJORAS

El trabajo desarrollado en este libro es solo una parte de muchas posibles mejoras e implementaciones. Se hacen las siguientes propuestas como extensiones posibles a este trabajo:

- Interfaz grafico con pantallas táctiles para la rapidez en el manejo del dispositivo, que permita ver a futuro los comportamientos deseados de la respuesta del sistema luego de acoplarlo al controlador inteligente.
- Optimización de la velocidad de respuesta del algoritmo de bacterias.
- Mayor capacidad para el número de dispositivos conectados al controlador inteligente.
- Fusión de técnicas de control junto con las técnicas ya definidas en el controlador inteligente.
- Mayor numero de interfaces de voltajes para sensores cuyas especificaciones técnicas no están dentro del rango de 0-5V.
- Uso del ADC de 20bits dentro del PSoC para sistemas de respuesta lenta y en los que se requiere una mayor precisión.
- Uso de los Filtros Digitales dentro del PSoC para evitar carga en la CPU con cálculos que siempre se estarán ejecutando en paralelo.

- Hacer inteligente la selección de parámetros necesarios para el algoritmo de bacterias como la selección de la población variables que permiten la exploración y explotación de la función de costo propuesta.

- Implementar más modelos para la función de costo además del ARX.

BIBLIOGRAFÍA

[Adex, 2010] Control Adaptativo Predictivo, Adex [en línea] [Consultado 10 de Abril de 2010]. Disponible en Internet: http://www.adexcop.com/es/tecnologia/control-adaptativo-predictivo.html.

[Aguilar, 2007] AGUILAR, J. and LABRADOR, A., Un Algoritmo de enrutamiento Distribuído para redes de comunicación basado en sistemas de hormigas, IEEE Latin America Transactions, vol. 5, no. 8, Dec. 2007.

[Aragón, 2006] ARAGÓN, V., CAGNINA, L., and . ESQUIVEL, S, Metaheurísticas basadas en Inteligencia Computacional Aplicadas a la Resolución de Problemas de Optimización Restringidos, Laboratorio de Investigación y Desarrollo en Inteligencia Computacional (LIDIC).Universidad Nacional de San Luis, Argentina, 2006.

[Ashby, 2005] ASHBY, Robert, Designer's Guide to the Cypress PSoC, ISBN-10: 0750677805, [Consultado el 10 de Abril de 2010]

[Blissett, 2002] BLISSETT, L. Código Abierto y Bacterias, 2002.

[Cypress, 2010] PSoC 5 Architecture, Cypress Semiconductor [en línea] [Consultado el 10 de Abril de 2010] Disponible en Internet: http://www.cypress.com/forums/forum,

[Darukur, 2008] Darukur , Curso de RTOS con FreeRTOS [en línea], Disponible en Internet: http://sistemasembebidos.com.ar/foro/index.php?topic=237.0

[Deb 1995] DEB, K., Optimization for Engineering Design: Prentice-Hall,1995.

[Dorigo, 2004] DORIGO, M. and STUTZLE T., Ant Colony Optimization. Mit Pr - Estados Unidos, 2004.

[FreeRTOS, 2011] The FreeRTOS Project, [en línea], FreeRTOS.org, Disponible en Internet: http://www.freertos.org

[Gomer, 2006] GOMER, J., Inteligencia Colectiva. Swarm Intelligence, Seminario LISI. Departamento de Ingeniería de Sistemas e Industrial. Universidad Nacional de Colombia, 2006.

[Gomez, 2011] GÓMEZ, Juan Carlos, Curso de Identificación de Sistemas [en línea], 2011, Disponible en Internet: http://www.fceia.unr.edu.ar/isis/

[Herrera, 2008] HERRERA, F, Computación Evolutiva, Dpto. Ciencias de la Computación e I.A. Universidad de Granada. España, 2008.

[Jiménez, 2000] JIMENEZ, J. A. Alonso and GUTIERREZ, M. A., Inteligencia Computacional y Conocimiento, Dpto. Ciencias de la Computación e inteligencia de artificial. Universidad de Sevilla, 2000.

[Marulanda, 2007] MARULANDA, Juan F., Control Inteligente De Un Reactor Químico, Grupo de Percepción y Sistemas Inteligentes, Universidad del Valle, 2007.

[Meyer, 2006A] MEYER, J. R. Entomología General de las Hormigas – Entomology [en línea], Department of University, NC State, 2006, Jan, Disponible en Internet: http://www.cals.ncsu.edu/course/ent425/tutorial/Social/ants.html

[Meyer, 2006B](--------), General Entomolgy Social Bees - Entomology, Department of University, NC State, 2006, Jan, Disponible en Internet: http://www.cals.ncsu.edu/course/ent425/tutorial/Social/bees.html

[Muñoz, 2008] MUÑOZ, M. A., LOPEZ, J. A, and CAICEDO, E. F., Inteligencia de enjambres: sociedades para la solución de problemas (una revisión), REVISTA INGENIERÍA E INVESTIGACIÓN, vol. 28 no. 2, p. 119--130, Agosto de 2008.

[Muñoz, 2007] --------, Optimización de funciones inspirada en el comportamiento de búsqueda de néctar en abejas, Memorias del Congreso Internacional de Inteligencia Computacional (CIIC2007), 2007.

[Oppenheim, 1998] OPPENHEIM, Alan V., Señales y Sistemas 2ed, Pearson Educación, 1998.

[Passino, 2010] PASSINO, K.m. Bacterial Foraging Optimization, Int. J. Swarm Intelligence Research, 2010.

-------- Biomimicry of Bacterial Foraging for Distributed Optimization and Control: IEEE Control Systems Magazine, June, 2002, vol. 22, no. 3, p. 52-67.

[Reyes 2006] REYES, J. V., Evolución diferencial para problemas de optimización de espacios restringidos. Master's thesis. México: Sección de Computación, CINVESTAVIPN, 2006.

[Wikipedia, 2010] Wikipedia: la enciclopedia libre [en línea]. Algoritmos Evolutivos, 2010, [Consultado el 10 de Abril de 2010] Disponible en Internet: http://es.wikipedia.org/wiki/Algoritmo_evolutivo

[Wikipedia, 2010A] Wikipedia: la enciclopedia libre [en línea]. PSoC, [Consultado el 10 de Abril de 2010] Disponible en Internet: http://es.wikipedia.org/wiki/Psoc,

[Wikipedia, 2010B] Wikipedia: la enciclopedia libre [en línea]. Inteligencia de enjambre, [Consultado el 10 de Abril de 2010] Disponible en Internet: http://es.wikipedia.org/wiki/Inteligencia_de_enjambre,

[Ziad, 2009] ZIAD, S.,The Bees Algorithm and this applications, Masaryk University, Brno, Czech Republic, Apr. 2009.